

[THE GIER ALGOL COMPILER GIER VERSION]

[Definition of storage of compiler]

```

b c100, e100           ; Begin of outermost block
d c70=39               ; First track during loading of compiler
d c60=39               ; After loading the compiler will be moved to have
                        ; its first track in c60
d e86=0                ; Reserve e86 tracks after top of translator
d e20=319              ; Last track used by translator
d e96=0                ; Reserve track for HP-entry
d e97=319              ; Last track accessible to todrum and fromdrum
d e68=0                ; if UV double then 1 else 0
s                     ; Allow manual redefinition of c70,c60,
                        ; e86, e96, e97, and e20
d e70=c70-c60          ; Resulting displacement of translator
d e85=0                ; Reserve e85 tracks after standard procedures
d e99=e96+e96           ;
d e99=e99+e99           ;
d e99=e99+e99           ;
d e99=e99+e99+e99+e99+e99 ; e99:= 40xe96
d e69=0                ; if CDC then 1 else 0
d e13=i+39             ; First core cell +39
d e25=983              ; Last core cell -40
d e4=e13+194           ; Place for universal translator parameters
d i=0, e90=k           ; First track of core picture
d e89=205e13           ; Standard procedure list[0]
s

```

Running system 1, track c61=c64-4

In addition to the code below the following words have been initialized by pass 8 and this track:

```

0:    qq  _1 , hv c17 ; last track of program, overflow alarm (spill)
3:    vk  6c61 , lk 1c ; call CHOICE OF
4:    vk  (1c62) , hv 17 ; OUTPUT UNIT
c:    qq 1c62.29-1.9, hs s2 ; priority word for track place 1
41c:  qqf c62.29 , hsf s2 ; - - - - - 2 ]
b k=c60+e70, i=822, a30, b20 ; begin running system block
d c61=c60 ; first track of running system
d c=6 ; first track place at run time
      arn  0 , ga r1 ; DISPLAY. Initialized from track 5c61.
d i=i-e68 ; Adjust for UV double
      arn  0 , ga r1 ;
      arn  0 , ga r1 ;
      arn  0 , ga r1 ; The following code is part of the initi-
      arn  0 , ga r1 ; alization for run. It is called from 5c61.
c94:lk 42c , vk (0) ; Last program track to core;
      pa  c t -1 ; Adjust priority word c
      pm 42c , gm a22 ; initialize start of program;
      pm 43c , gm a23 ; initialize drumplace;
      arn 44c , ga a21 ; initialize last used;
      pm r1 , hv s2 ; prepare initialize display; return;
      arn  0 , ga r1 ;
      qq  0 , hv a17 ;
e79: c1:qq  0 , pp  0 ; display block 0; working word output and
      ; write;
      [working storage, constants]
c33:c35:qq -2.9 + 512.19-1.39 ; a half, close enough
      c34:qq 1.39 ; eps
      c37:qq [fullword] ; UV
      qq ; if e68=1 then
d i=i+e68-1 ; insert UV 2
      c40:qq 0 t 256 ; 1.0
      c41:qq -1 t 512 ; -1.0
      c43:qq 511 t 510 ; nonsense
e82:c46: ;
      a21:qq 0 ; last used, initialized above
a22:c19:zq , zq ; start of progr., above: hs c3,track no. rel.
      c47:qq 0 ; appetite, working address, bits 10-39 = 0
      c48:qq 0 ; input case
      c49:qq 0 ; last character including case
      c56:qq 41.10 ; constant for counting top of program etc.
      c57:qq 801.19+2.39 ; constant for generating priority word
      c58:qq -1.19-50.25+60.35 ; constant for generating sr-word
      c63:qq 0 t 0 ; input checksum
      c68:qq [fullword] ; working location for standard procedures
e80:c69: ;
a23: qq [fullword] ; drum place (floating representation), above
      c71:qq -1.9+1.11 ; a half, exactly (entier)
      c82:ps 0 t 11 ; working word for output and write
c91:c89:qq s1 , vy 32 ; char is set (false,true=s35),
      ; medium output and input
c92:c90:qq 511 , vy 17 ; min (drum place); medium write and type
      c17:ps 7 , hh a18 ; go to overflow alarm (entry from 0)
      [tracktable]
      ca  0 , hh 779c ; track - - is in trackplace 779 c
      ca  0 , hh 738c ;
      ca  0 , hh 697c ;
      ca  0 , hh 656c ;

```

[2.9.65]

[Running system 2, track 1c61=c64-3]

[track table continued]

```
ca 0 , hh 615c ;
ca 0 , hh 574c ;
ca 0 , hh 533c ;
ca 0 , hh 492c ;
ca 0 , hh 451c ;
ca 0 , hh 410c ;
ca 0 , hh 369c ;
ca 0 , hh 328c ;
ca 0 , hh 287c ;
ca 0 , hh 246c ;
ca 0 , hh 205c ;
ca 0 , hh 164c ;
ca 0 , hh 123c ;
ca 0 , hh 82c ;
c62: a4: ca 0 , hh 41c ;
ca 0 , hh c ;
```

[call track from drum, track table lookup will go on here when track is not in]

```
ga b1 MA ; set current track;ensure mark B inR.
a9: hsn a1 , ps s-41 ; get address of last priority word
b2: arn 0 , sr s ; trackstart. get address of lowest
gs r-1 NT ; priority and store it in trackstart b2
hh r-3 LB ;
b1: vk 0 , ps (b2) ; current track. Read track to
lk s1 , arn s ; trackplace
ck 10 , gt r1 ; set current track in
it (b1) , pa 0 ; track table
arn c46 , ck -1 ; Test if new track place can
sr b4 , sr c56 ; be reserved.
hv a2 LT ; no.
arn c57 , hs a1 ; Get address of new trackplace
c93: b4: qqf 84c.10 ; top of program.10
ar (b5) D -1 ; and set new priority word.
ck 20 , pa (b5) ; Format of priority word:
gr s41 , arn a3 ; priority, 2, corresponding
ga s41 MC ; addr in tractable, hs s.
arn c56 , ac b4 ; top af program:=top of program+41
a2: vk (b1) , hv a20 ; wait for drum; goto set s to trackstart
a5: ga 41c , ps r ; clean up:
a1: hh (b5) , ps s-41 ; get address of last priority word:
ann s , tk -8 ; s:=addr of last priority word
ga s , ca 0 ; ZB:=false;
```

[6. nov. 1963]

[Running system 3, track 2c61=c64-2]

```
qq    (b5)    V  1      NZB ; repeat: priority[s]:=priority[s]:256
pan   a3      V  1      IZB ;           if priority[s]=0^-,ZB then
arn   c56     ,  sc  b4   ;           release trackplace else ZB:=true;
hh    a1      LB      ;           s:=s-41; if B then goto repeat;
c27:  zq      LKA      ; CHOICE OF OUTPUT UNIT, hv3LKA or hv3NKA
a20:c28: ps   (b2)    ,  hv  a3 ;           Prior:=1; goto exit track adm.
```

[entries to track administration]

```
c2:   qq    p0      IRC ; transfer track P: p0 used by output and write
gm    b6      MRC ;           save M and marks
a17: c3:  pm    s      XV IRC ; goto track P: M:=R; R:=parameter[s]
c42:  pm    b4     ,  ud  a19 ; call track R: indicate not transer set ex.
c4: c21:  gt    b7     ,  ck -10 ; goto track R: set reladdr, Raddr:=trackno.
b5:   hs    a4     ,      ;           track table look up; may return s2
c53: b6:   qq    0     t  0      ; working storage for save M
ga    b1     ,  gs  b2      ;           set current track, set trackstart
a3:   arn    0     D  1      IZB ; exit trackadm:count priority; OB := 0
ga    s      XV      NO      ;           set priority
ga    s      ,  hv  a5      ;           priority overflow, goto clear up
b7:   pm    b6     ,  ps  s0    ;           restore M; s:=s+rel addr
hh    s1      LRC      ;           exit to track
hv    s1      LRA      ;           exit to track
b15:  hv    0     t  1      ;           exit from take track
c65:  gm    b6     ,  gs  b15    ; call track P: save M; set exit
pmf   r      X      IRC      ;           indicate not transfer
arn   s      ,  hv  c21      ;           parameter; goto goto track R;
```

[Initialize block or call]

```
c5:   it   (c46) ,  pa  b12    ; Init.block or call: descraddr:=last used
pmn   s      X      IRC      ;           get parameter
ck    -10    ,  ga  c47      ;           set appetite
ar    c46    ,  ga  c46      ;           lastused:=lastused+appetite
hv    a7      LRB      ;           if initialize block then
tk    10     ,  gt  b9      ; block: begin set display address
ar    c58    ,  pm  c46      ;           (blockno); generate sr word with
gr   (b12) X  -2      MA      ;           old sr; descraddr:=sr:=descraddr-2
gp   (b12) ,  pp  (b12)      ;           store sr word
b9h:  gr    p1     ,  gp  0      ;           store lastused; set display
a7:   ck    -1     ,  sr  b4    ;           end; if last used-top of program ≥0
hv    a6      NT      ;           then goto param transfer;
hs    a8      IRC      ;           release track places
pm    s2     DX      ;           if current track place is not
ck    -1     ,  sr  b4      ;           released then
hv    a6      LT      ;           goto param transfer
```

[6. nov. 1963]

[Running system 4, track 3c61=c64-1]

```
a19:  pa  b15  t  b12  IRC  ;           else call current track to
      nt  s    ,  nt (b2)  ;           new trackplace and return
      pt  b7   ,  hv  a9   ;           to param transf

      [index call, parameter transformation]
c26:  gm  b13                ; index call:  array ident. in M, index func in
      sr  (b13) t  -2        ;           R:=R-constant term
      sr  (b13) V  1        NT ;           if R>0 then R:=R-length
      ps  6    ,  hh  a18    ;           else index alarm
      ck  -10   V          LT ;           if R<0 then shift to Raddr
      ps  6    ,  hh  a18    ;           else index alarm
      ga  b13                ;           UA:=Raddr
c36:b13: qq  0                ;           UA:=UA+baseaddr; return
c20h:  hv  s1   ,  ck  10    ; local switch: positions to R addr
      ga  r2   ,  tk  -30   ;           positions+1 to R39; NB:rounding
      ar  (b12) D          IRA ;           Raddr:=descradr+
c80:  ar  0    DV          ;           positions; skip next
      a11: ar  p    D          ; add sr:
c81:  b12: gr  0    V  -1    MRC ; store descr: count descradrandstore; skipnext
      a6:  gs  r1                ; paramtransf: set paramaddr
c84:  pmn  0    X  1    IRC  ;           R:=next parameter
      hv  b12                NA ;           constant
      hv  a11                NT ;           programpoint
      gr  r1   ,  pm  r-1    ; the rest:  prepare execute parameter
c54:  qq  0    ,  qq  0    ;           execute parameter
      pm  s    DV          LRC ; if simple then M:=abs addr (simple var);
      hh  a14  X          LZ  ; if proccall then goto enter proc
      hv  b12  X          IRC  ;           goto store descr

      [reserve array]
c50:  arn (s)                ITA ; reserve array: TA:=length<0
      arn  c46 ,  ck  10    ;           R39:=lastused-length
      sr  (s)                ITB ;           TB:=R < 0
      ck  -10   V          NTC ;           if -, (TA<TB) then R39 to Raddr
      ps  4    ,  hh  a18    ;           else array alarm
      ga  c46 ,  ga  p1    ;           lastused:=stack[sr+1]:=lastused
a16h: hv  a7   ,  ar  c56    ;           goto test last used; release 1:
      pm  c56  X          ;           R:=R+41; top of program:=
      sc  b4   X          ;           top of program-41;
      hr  s1   ,          NT ;           if R>0 then return s1
      a8: it  (b5) t  1    ; releasetrackplaces: if decrease trackplaces >2
      bs  1a4 ,  hh  a16    ;           then go to release 1 else alas alarm
c55h:a18h:ps  13 ,  vk  2k-e70 ; alarm: call first track with alarm
      lk  a15 ,  vk  2k-e70 ;           number in s
```

[6. nov. 1963]

[Running system 5, track 4c61=c64]

d c64=k-e70 [free trackplace starts in c67, 40 instructions from here
are stored on track c64]

```
a15:c38: c67: hv    a12      LZ ; value switch des: if index in R39 ≤ 0 then
             hv    a12      LT ;                               goto outside range
             gm    c37 , it (c37) ;                               UV:=switch identifier:=M
             pa    r3 , pa  c37 ;                               set base for switch
             sr    c37      ITA ;                               R 39:=R39-positions-1
             ck    20 , gt  r1 ;                               TA:=R<0; if TA then
c85: pm    0  V  0      LTA ;                               M:=stack [base+index-positions-1]
a12: pmnf  r    X      ; else outside range:M:=0; marks:=B;
c22: pa    b13 t  c37  IRC ; take formal:UA:=address(UV);
             gm    b13 V    NRC ;                               if simple variable then UA:=M
             gm    c37 X    MRC ;                               else UV:=M
             hv    s1      NRA ;                               if -, programpoint then return
c9:a13: pm  (b1) DX      IZB ; set return: lastused:=lastused-1
             pm    c46 , ck  10 ;                               stack [last used]:=pack
             gr    (c46) X -1  MA ;                               (sr, s-trackstart, 0, trackno)
             nt    s , nt  (b2) ;
             pt    (c46) , gp (c46) ;
             ck    -1 , sr  b4 ;                               R:=lastused+1-top of program
             hs    a8      LT ;                               if R<0 then release trackplaces
             hv    c3      LZB ;                               if standard proc then goto goto track
a14h: ps    c37 , arn s ; else s:=addr(UV); update: R:=param
             pp    (s) , hh  p ; sr:=sr part (param); switch to proper display
c10: grf  (c46) V -1  MB ; call st proc 1 fl value:
c11: gr  (c46) t -1  MB ; call st proc 1 nonfl: R:=0 goto set return
c7h: hvn  a13 , pm  p-1 ; exit func: UV:=func value
             pa    b13 t  c37 ;                               UA:=address(UV)
c6h: gm    c37 , gs  c46 ; exit proc: last used:=s
             ps    p2 , hh  a14 ;                               s:=sr+2; goto update
c18: hv    s1      LZ ;goto computed: if labelinR=0 then return; s:=UA
c16h: ps  (b13) , pp  (s) ;                               goto non local: sr:=srpart (label)
             it  (p1) , pa  c46 ;                               lastused:=stack [sr+1]
             arn  s , hh  p ;                               update and transfer

c13: grf  c37 V      ; exit RF expr: UV:=RF; skipnext
c14: gr    c37      ; exit R  expr: UV:=R
c12: pa    b13 t  c37 ; exit UV expr: UA:=addr (UV);
c15: ps  (c46) , it  1 ; exit addrexp:s:=last used; t:=1; skip next
c51: qq  (c46) t  2 ; exit st proc 1V:t:=2, lastused:=t+lastused
c83h: hh  a14 , pp  0 ;                               goto update;

c8: pp    p2 , gp  c46 ; exit block : lastused:=sr+2
     pp  (p-2) , hv  s1 ;                               sr:=stack [sr]; return
```

[vk first track, hv 0 ; this last instruction is set by the program
start track, the last instruction on the last track of the running
system being the preceding instruction]

d c60=7c60

e

[20.Dec.63]

[Running system 6, track 5c61]
[Error printing and program start.]

b i=c67, a20, b10

```
a16: gs   a18 , hs   a17   ; type text indicated by s
      vk   7c61 , ps   10   ;
      lyn  40 , ca   41   ; input from typewriter
      gs   a18 , hv   a19   ; if symbol = r then go to a19
      lk   40 , vk           ; transfer track 7c61
      hv   40                ; go to error analysis or compiler ready
b7:   qqf c62.29 , hsfs2 ; initial value for 41c
b9:   vk           , hv   c25 ; initial value for 40c67
a17: vk   6c61 , lk   1c   ; running system 7 to first track place
      vy   17 , syn  64   ; text printing routine entry
a18: arn r10 t   1   LZ   ; address of text in address part
      ca   10 , hv   s1   ; if character = 10 then exit
      ga   r1 , tk   10   ;
      sy   0 , ck   -4   ;
      hv   a18           ;
```

[texts:]

```
[4] qq 49.9+41.15+41.21+49.27+24.33+10.39 ; array (c50)
[5] qq 58.9+35.15+37.21+10.27             ; ln
[6] qq 57.9+37.15+52.21+53.27+23.33+10.39 ; index (c26, twice)
[7] qq 18.9+39.15+57.21+35.27+35.33+10.39 ; spill (c17)
[8] qq 58.9+18.15+40.21+41.27+19.33+10.39 ; sqrt
[9] qq 58.9+53.15+23.21+39.27+10.33       ; exp
[10] qq 58.9+41.15+20.21+37.27             +10.39 ; run
[11] qq 58.9+53.15+37.21+52.27             +10.39 ; end (pass 8, 1st order in output)
[12] qq 58.9+52.15+41.21+20.27+36.33       ; drum (to drum)
[13] qq 58.9+49.15+35.21+49.27+18.33+10.39 ; alas (a8)
[14] qq 58.9+55.15+38.21+37.27+53.33+10.39 ; gone (above)
[15] qq 39.9+49.15+41.21+49.27+36.33+10.39 ; param (standard procs more param)
```

```
c23:a19: vk   c61-1 t 1   ; transfer
      lk c67-200 t 40     ;   tracks 1-4
      bt   3   t -1      ;   of running system
      pm   b7   , hv   a19 ;
      gm  41c   MC       ; initialize 41c
      vk   (0) , hs   c94 ; call initialize run on c61
      gm  c1-1 t -1 MA    ; initialize display
      nc  (r-1) , hv   r-1 ;
      pm  c19 , hs   a17 ; type: run
      pan c24 t  c19     ; set exit from CHOICE AF OUTPUT UNIT
      cl  -10 , ar   b9  ;
      gr  r2   MA       ; initialize 40c67
      vk  c64 , lk   c67 ; transfer last track of running system
[40c67:vk first track, hv 0 ]
```

e

[6.9.65]

[Running system 7, track c66]

[Core store 0-6, CHOICE OF OUTPUT UNIT, entry from SLIP]

[This track is placed from 0-39 by pass 8 or by reading a binout tape
and later in the first track place 1c-40c by 5c61 and on jump to 3 from c27]

b i=c67-40,a20

```
[0] a: qq 0 , hv c17 ; last track set by Pass 8,3a69, jump on overflow
[1] vk e14 , lk c67 ; call INITIALIZE
[2] vk c61 , hv c67 ; TRANSLATOR
[3] vk 6c61 , lk 1c ; call CHOICE
[4] vk (1c62) , hv a5 ; OF OUTPUT UNIT
[5] lk 1c , hv 40c67 ; return to running system
```

d i=c+a

```
a3: qq 1c62.29-1.9 , hs s2 ; initial priority and reference
a1: hv 3 NKA ; for track place
a2: qq 1.34 ;
[a5:c25:]a4:vy 17 , arn ral ;
ar ra2 NKA ;
gr c27 , lyn ral ; set conditional jump
ar ral0 , ca 48 [e] ; if symbol = e then go to slut;
ps 11 , hh c55 ;
ca 17 [w] , hv ra9 ; if symbol = w then write output
nc 34 [p] , ar ra8 ; else if symbol = p then punch output
nc 49 [b] , ck 20 ; else if symbol = b then all output
a9: pt 40c67 t c28 ; else normal output;
gt c89 , ud c89 ;
ck -10 , gt c90 ;
[c24:] a7: hv 5 ; first time hv c19
a10:qq -5.9+16.19+33.29+32.39;
a8: qq 4.9+32.19-16.29 ;
a21:psn e87 , vk s c61-1 ; entry from SLIP for checksum
a14:lk (ral1) , pp 40 ;
ps s-1 , vk s c61-1 ; exchange buffers
it (ral1) , pt ral3 ; Communication with SLIP for entry
a11:nt a16 , lk a17 ; by h algol:
a12:pp p-1 , it -512 ; 6c61 , e95 = track , relative for
a13:bs sc61-550, ar p-1 ; entry from SLIP;
bs p , hv ral2 ; a18 and a20 error exit and parameter.
bs s , hh ral4 ;
ck 0 ; a16 buffer area begin
hv 1 LZ ; a16+80 buffer area end
ps a20 , hv a18 ; track 38 is omitted during summation
qq ; if sum ok then goto 1
c29:pa 0 t e20 ; else go to error;
vk 5c61 , lk c67 ; entry from binout reading:
vk 5c61 , hv c23 ; go to start run on track 5c61;
a19:qq ; sumbalance for this track
```

```
d a16= c67-160 ; buffer 1 (first address of running syst)
d a17= a16+a16+40 ; buffer 2
d a18=626 ; error exit to SLIP
d a20=714 ; parameter for SLIP
d a5 = a4-a+1a3-a ; entry when placed in 1c
d c25 = a5 ; - - - -
d c24 = a7-a+1a3-a ; - - - -
d e95 = a21 - a ; entry from SLIP
d c66 = 6c61 ; this track
d c29 = c29 - a ; relative entry from binout reading
```

e

d i=10 ; for loading purposes only

s

[6.9.65, Running system 8, track 7 c 61]

[This track is called by 5 c 61 and placed in core store 42c-81c. It calls the program for analysis and output of the stack (part 1, 2 and 3)]

b i = c67-80, b30, a5;

```

        it(c62+7), pt c34 ; save current track-number
        qq 2 , hsr b 2 ; transfer to part 1
b1:  qq 3 , hsr b 2 ; transfer to part 2
b3:  qq 3 , hsr b 2 ; transfer to part 3
b4:  vk 5c61 , lk c67 ; call track with error printing
b0:  ps 11 , pm rb13 ; go to printing of ‡<end‡
        gm c92 , vk 0 ; after output of stack Restore min drumplace in c92
        qq e84 , hv c67 ;
b2:  pa r 2 , tc67-160 ; subroutine for transfer
        vk 7c61 t 1 ; of program for analysis
        lk t 40 ; and printing of stack
        bt (s) t -1 ;
        hv r-3 , ;
        vk 0 , hvc67-120; exit from subroutine
b5:  ptr a 0 , mln ral ; subroutine belonging
b6:  bt IZBt-1 ; to part 3: printing
        hhr b 5 , ; of M regarded as
        ck -10 , gar a 2 ; decimal fraction
        hvr a 4 LZ ; with sign in b8
a3:  ptr a 0 t 16 ; An optional number
        hvr a 0 NRA ; of digits in the
        arnrb 8 , ca 0 ; beginning are
        sy 0 , hhr 1 ; discarded an
        sy 32 , ; optional number
        hvr a 0 IRA ; are printed
a4:  can(rb7) , hvr a 3 ;
a0:  par a 2 LOBt 0 ; Output of sign
a2:  sy , ; only if RA is set to 1 before entrance
b7:  bt t-1 ;
        hhr b 5 , ;
        zq s 1 ; exit when KA not changed
        hv 9c ; exit when KA changed
b8:  qq 0 ; e1 in RS 9-11
b12: ;
a1:  m ;
        10 ;
b9:  qq e81 , ; (last track number of standard procedures)×2‡(-10)
b10: qq e20 ; (total number of tracks)×2‡(-10) RS9:6a13,RS10:a4-6
b11: qq e 0 ; (first track number of standard procedures)×2‡(-10),RS10
b13: qq ; min(drumplace)
b14: qq ; 40×total number of tracks×2‡(-10),RS12:2a0,RS9:8a13
b15: qq , qq ;
d a5=c67-c-122 ;
d a5=c67-120 ;
d b0=b0-a5, b1=b1-a5, b2=b2-a5 ;
d b3=b3-a5, b4=b4-a5, b5=b5-a5 ;
d b6=b6-a5, b7=b7-a5, b8=b8-a5 ;
d b9=b9-a5, b10=b10-a5, b11=b11-a5 ;
d b12=b12-a5, b13=b13-a5, b14=b14-a5 ;
d b15=b15-a5 ;
```

[6.9.65, Running system 9, track 8 c 61]

[The program for analysis of return information consists of this track and tracks 9 - 10 c 61 referred to as part 1 in the comments on track 7c 61]

b k = 8c61+e70, i= c67 - 120, a19, d9, e5

d d0=c47,d1=26c,d8=27c, d7= 28c, e5=i+120, e1= b8, e2= 30c

```
pm c92    , gm  b13    ; save min(drumplace)
ga  e4    , vk  5c61    ; save input symbol
pm a17    IRC1         ; move subroutines and constants
gm  2c    MRC1         ; used by part 3 and 1
arn r-2   , nc  a18     ;
grn d7    , hv  r-3     ;
arn e4    , hs  10c     ; go to examine input symbol
hv  a13   LZ          ; R=0 if input symbol =p,w or b
arn(2b0)  D           ;
ck  -1    , sr  c92     ;
lk  c67   , vk  0       ; call track with error printing
hv  1     LT          ; go to pass 1
ps  14    , hv  c67     ; if compiler is gone go to alarm
a13: ps (c1) , arn c46   ; if no stack go directly to printing of {<end>
ca  s2    , hv  b4      ;
arn( b9) D-1          ; calculate constants
ck  -1    , gr  b9      ;
arn(b10)  D1          ;
ck  -1    , gr  b10     ;
ck  -2    , ar  b10     ;
ck  -1    , gr  b14     ;
arn b11   , ck  -1     ;
gr  b11   , srn c34     ;
bs (c43) , hv  a19     ;
pa  b3    t 3         ; if stack already analyzed go directly to print
pa b2+1   t 14c61     ;
grn d8    , hv  b3      ;
a19: tk  9    , ar  b9   ; if current track< last track number of standard
hv  a11    LT         ; procedures then begin expecting call return:=true;
pp (c46) , ppn (p)    ; sr:=part1(stack[last used]) end else
a11: ps (c46) IZA      ; expecting call return:=false
a1:  ps s-1   , arn p1  ; last used:= last used - 1
ck  -1    , gr  d8     ; new sr: last used for local block:= stack[sr + 1]
a2:  ps s1    , gs  d0   ; next word: last used:= last used + 1
arn d0    , ck  -1     ;
gr c37    , sr  d8     ; if last used>last used for local block
hh  a3     NT         ; then go to block found
grn e2    , arn  s      ;
hv  a2     NA         ; if -, mark a[last used] then go to next word
ga  e1    , ck  10     ; stack reference:= part1(stack[last used])
```

[Running system 10, track 9 c 61]

```
ga e2 , ck 10 ; track relative:=part2(stack[last used])
nc 0 , hv a2 ; if part3≠0 then go to next word
ck 10 , ga e 4 ; track number:= part 4(stack[last used])
arn e 1 , ck -1 ;
gr d 1 , arn c37 ;
sr d 1 , ; if last used ≥ stack reference then
hv a 2 , NT ; go to next word
arn e 2 , gp d0 ;
hv a 2 , LT ; if track relative > 511 then go to next word
sr 40 , D ;
hv a 2 , NT ; if track relative ≥ 40 then go to next word
arn e 4 , ck -1 ;
sr b10 , ;
hv a 2 , NT ; if track number>totalthen go to next word
ar b10 , sr b11 ; if track number < first standard then
hv a 2 , LT ; go to next word
hh a 4 , NOA ; if -,expecting call return then go to parameter call
acn s , MB ; marks[last used]:=01
a4h: hv a 5 , arn d1 ; go to P; parameter call:
qq 1 , gr 31c ;
can (e2) , hv a2 ; if track relative = 0 then go to next word
arn e4 , sr r-2 ;
ck -1 , sr b9 ; if track number≤ last track number of standard procedure
hv a15 , LT ; then go to S
e4: vk , lk e 5 ; transfer from drum(track number) to (buffer address)
vk , is (e 2) ;
arns+e5-1 , gr d 1 ; program word:=store[buffer address + track relative -
arn d9 , is (e2) ; if (track relative >37 ∨ track relative = 37 ∧
ca s-66 , sr 38e5 ; stack[buffer address + 38] = convert instructions)
it (e2) , bs 38 ; ∧ store[buffer address + 39] = transfer instruction
hh a6 , NZ ; then begin
arn 39e5 , gt r4 ; track relative := part2( store[buffer address + 39]);
ck -10 , ga a7 ; track number := part4( store[buffer address + 39])
mb d3 , sr d4 ; transfer from drum(track number) to (buffer address)
hh a6 , NZ ; end;
pa e2 , t ;
a7: vk , lk e5 ;
a6: vk , is (e2) ;
arns+e5 , gr e5 ; program word 1 := store[buffer address + track relative
arn d1 , mb d3 ;
```

[Running system 11, track 10 c 61]

```

      sr d6 ;
      hv a2 NZ ; if program word ≠ take formal then go to next word
      arn e5 , ck -10 ;
      mb d3 , sr d5 ;
      hv a8 NZ ; if program word 1 = value call then
a15: acn s MC ; begin S: marks[last used]:= 11;
a10: hvn a 9 IZA ; R: expecting call return:= true; go to Q end
a8:
a14: acn s M ; marks[last used]:=00;
a5: arn r IZA ; P: expecting call return:= false;
a9: arn d 7 , ck -20 ; Q: stack[last used]:= stack[last used] +
      ; return chain×2↑(-29);
      ar s , grs ;
      ck 10 , ca c1-1 ;
      hv b1 ; exit when outermost block is found
      pp (e 1) , gs d 7 ; sr:= stack reference; return chain:= last used
a3h: hh a 1 , ps p ; go to new sr; block found: last used:= sr
      acn s MA ; marks[last used]:= 10;
      arn s , ga e 1 ; stack reference:= part 1(stack[last used])
a17: hv a10 ; go to R;
      gr b8 , ann b8 ; subroutine for printing of integer in R
      nkf 39 , mkf c49 ; unit in position 39, is used by part 3
      tkf 10 ; after being moved o core store 3c-8c
      ar b12 X ;
      hs b5 ;
      hr s1 ;
      ly 17c ; program used by part3 wwhen KA is changed
      ga 17c , arn ra16 ; (core store 9c-18c)
      ar 2c LKA ;
      gr b7+2, arn 17c ; store actual value of KA in jump instruction
      ca 22 , vyn 17 ; input symbol = w
      ca 39 , vyn 33 ; input symbol = p
      ca 50 , vyn 49 ; input symbol = b
      ca 53 , hv b4 ; if input symbol = e go to print ‡<end‡
      qq , hr s1 ;
a16: hr s1 NKA ;
a18: m ;
      0.0005 ; round-off constant, stored later in 19c
d3: , udn(p-1) ; mask for left instruction
d4: , hs c2 ;
d5: , ps (c46) ;
d6: , hs c22 ;
d9: tkf -29 , nkf 39 ;
```

e

[7.9.65, Running system 12, track 11 c 61]

[This track and tracks 12-14 c 61 contain the analysis of local quantities referred to as part 2 in the comments to track 7 c 61]

b k = 11c61+e70, i = c67-120, a27, d6, e4

d d1= 31c, d2= 32c, d3= 33c, a10= 28c, a11=30c, d5=26c, e2= 29c

d e1= 27c, d6= d5

```
grn 34c    t -1      ;
arn  r-1    , nc    26c ; clear working locations for part 3
hv   r-2    ;
pm   e 4     IRC 1     ; move subroutines
gm   a25     MRC1     ; and constants
arn  r-1    , ncc67-121; used by part3
hv   r-3    ,        ;
a1: ps (c 1) , hv   a 0 ; stack reference:= stack[DISPLAY[0]]
    ca  s    , hv   a 0 ; procedure : if stack reference = last used then go to block
    arn s-1   ,        ;
    hv  a 0    LA      ; if marks[stack reference-1]  $\neq$  b
    hv  a 0    NB      ; then go to block
a0: pp s-1    , hh   a 0 ; j:= stack reference - 1; go to common
    pp s      , arn s+1 ; block: j:= stack reference; common:
    gr d 1    , gr d 2 ; first variable:= first point:= last used
    arn b14   IZA      ; among variables:= false
    gr d 3    , gp   a10 ; earliest known program point:= 40xtotal number of track
a2: arn r      IZB      ; no switch: in switch mode:= false
a3: snn r      ITA      ; next word: expecting array identifier:= false
a4: pp (a10)  , arn d 1 ; following array: if j = first
    ca  p     , hv   a 5 ; variable then go to next block
    pp p-1    , gp   a10 ; j:= j - 1
a6: arn p      IRC      ; same word; marks[j] to RC
    gr  p     M         ; marks[j]:= 00
    ga e 1    , tk    10 ; e1:= part 1(stack[j])
    ga e 2    , tk    10 ; e2:= part 2(stack[j])
    hv a24     LOA      ; if among variables then go to variables
    nc  0     , hv   a 7 ; if part 3 = 0 then go to no more points
    tk  9     , gr   e 4 ; e4:= part 4(stack[j])/2
    sr b10    ,        ;
    hv  a 7    NT       ; if part 4  $\geq$  total then go to no more points
    arn e 2    ,        ;
    hv  a 7    LT       ; if part 2 > 511 then go to no more points
    sr  40     D        ;
    hv  a 7    NT       ; if part 2  $\geq$  40 then go to no more points
    gs a11    , arn e 1 ;
    ck  -1    , gr   a13 ;
    arn a11    , ck   -1 ;
    sr a13     ITB      ; sign of (stack reference - part 1) to TB
    hv  a 8     NRA      ; if marks  $\leq$  1 then go to switch or variable
```

[Running system 13, track 12 c 61]

```
hh a 9      LZ      ; if part 1 = stack reference then go to program point
hv a12      LT      ; if stack reference < part 1 then go to switch element
a9h: hv a 7 , arn e 2 ; go to no more points; program point:
ck -4      , ar e 4 ;
ck -2      , ar e 4 ; point:= part4×40 + part 2
ck -1      , sr d 3 ;
hv r2      LZ      ; if point>earliest known program point then
hv a7      NT      ; go to no more points
ac d 3      , hv a 3 ; earliest known program point:= point go to next word
a8: ncn(e 2) , hv a 7 ; switch or variable: if part 2 ≠ 0 then go to
; no more points
srn p      , tk 30 ;
ar e 1      , ca p   ; if j = part 1 - part 4 then
hv a14      ,       ; go to switch identifier
arn p      , tk 30 ;
hv a 7      NZ      ; if part 4 ≠ 0 then go to no more points
hv a 7      LRB      ; if RB = 1 then go to no more points
a12: hv a 3      LOB      ; switch element: if in switch mode then go to next word
gpn d 5      IZB      ; in switch mode:= true
hv a 3      ,       ; element address:=j,go to next word
a7: pp (d 5)      LOB      ; no more points: if in switch mode then
pp p 1      , gp d 2 ; j:= element address; first point:= j + 1;
hvn a6-1      IZA      ; among variables:= true; go to same word;
a14: hv a 7      NRC      ; switch identifier: if marks = 0 then go to no more points
hv a 7      LTB      ; if stack reference < part 1 then go to no more points
hv a15      NOB      ; if -,in switch mode then go to a
arn d 5      , ck -1 ;
sr a13      ,       ; if part 1 ≤ element address then go to no more points
hv a 7      NT      ;
a15: acn p      MA      ; a: marks[j]:= 10
hv a 2      ,       ; go to no switch
a24: hv a 3      LRB      ; variables: if marks ≠ 0 then
hv a 3      LRA      ; go to next word
arn p      , tk 21 ; if part 3 ≠ 0 ∨ part 4 ≠ 0 then
hv a 3      NZ      ; go to next word
arn e 2      , ck -1 ;
gr a16      , arn a10 ;
ck -1      , sr a16 ; if j < part 2 then
hv a 3      LT      ; go to next word
pp (e 1)      , arn p-1 ; if -,expecting array identifier then
hh a19      NTA      ; begin length:= stack[part 1 - 1]
```

[Running system 14, track 13 c 61]

```

      hv a 3      LT      ; if length < 0 then go to next word;
      gr e 4      , arn a10 ; if part 1  $\neq$  j + 3 then go to next word;
      ar 3        D      ; end else
a19: hv a20      , arn d 6 ; if part 1  $\neq$  length address + 1
      ar 1        D      ; then go to next word;
a20: nc (e 1)    , hv a 3  ;
      srn 90      D      ; if part 2 - length < 90 then
      ar e2      , ck -30 ; go to next word
      sr e4      ;
      hv a3      LT      ;
      arn e 2     ; if -,expecting array identifier
      ga d 1      LTA     ; then first variable:= part 2
      acn(a10)    MC      ; marks[j]:= 11
      hv a 4      NTA     ; if expecting array identifier then go to following array
      acn p-2     MA      ; marks[part 1 - 2]:= 10
      arn p-1     ITA     ; expecting array identifier:= true
      nkf 39      ,      ;
      grf p-1     MA      ; marks[part 1 - 1]:= 10
      pp p-1     , gp d 6 ; coefficients address:= length address:= part 1 - 1
a17: pp p 1      , arn d 2 ; new coefficients: coefficient address:= coefficient address
      ca p        , hv a 4 ; if coefficient address = first point then
      ; go to following array;
      arnfp      ,      ;
      hv a 4      LA      ; if marks[coefficient address] $\neq$ 00 then go to following array
      hv a 4      LT      ; if stack[coefficient address]<0 then go to following array
      hv a 4      LZ      ; if stack[coefficient address] = 0
      ; then go to following array
      arnf p      , srf (d6) ; if stack[length address] $\leq$  stack[coefficient address]
      hv a 4      NT      ; then go to following array
      arnfp      , tkf -29 ; if entier(stack[coefficient address])
      nkf 39      , srf p   ;  $\neq$  stack[coefficient address] then
      hv a 4      NZ      ; go to following array
      arnf(d6)    , dkf p   ; new length:=
      grf a13     , tkf -29 ; stack[length address]/
      nkf 39      , srf a13 ; stack[coefficient address];
      hv a 4      NZ      ; if entier(new length)  $\neq$  new length then go to following array
      arnf(d6)    , srf a13 ; if new length = stack[ length address ]
      hv a 4      LZ      ; then go to following array
      arnf a13    , grf(d 6) ; stack[length address]:= new length
a26: acn p        MA      ; accept: marks[coefficient address]:= 10
      hv a 17     ,      ; go to new coefficient
a5:  pp (d5)     , pp p1   ; next block:
```


[6.9.65, Running system 16, track 15 c 61]

[This track and tracks 16-18 c 61 form the program for output of the stack referred to as part 3 in the comments to track 7 c 61]

b k = 15c61 + e70, i = c67 - 120, a75

```
pm a60 , gm b23 ; word restored in case of repeated printing
sy 58 , hs b16 ; tryktekst(⟨<stack⟩)
qq 18.9+19.15+49.21+51.27+34.33+10.39;
a74: pp (c1) , sy 64 ;
arn p3 , sr b21 ; print own variables
hv a73 LZ ;
ga 27c , sr 27c ;
hv a71 LZ ;
arn a74 , ;
ca p3 , hv a71 ;
, ca p4 ;
ncn(c1-1), hv a71 ;
a73: arn p2 , hs a15 ;
pp p 1 , hh a74 ;
a71: pp (c1) , hv a2 ; sr:= part1(DISPLAY[0]); go to test;
a1: arn p1 , gr 30c ; follow chain: previous last used:= stack[sr+1];
a72: arn p , ck 20 ; a: sr:= part 3(stack[sr])
ga 28c , pp (28c);
ncn p , hv a2 ;
hs b16 ;
qq 18.9+19.15+49.21+51.27+34.33; if sr=0then begin tryktekst(⟨<stack_end⟩);
qq 53.15+37.21+52.27+10.33 ; go to finished end
hv b4 ;
a2: arn p1 , pm p ; test:
hv a 5 NA ;
hh a 6 LB ; if marks[sr] = 10 then begin
tk 10 , gr 30c ; first var:= part2(stack[sr+1])
tk 10 , gr 29c ; first point:= part3(stack[sr+1])
a6: hh a 9 , hs b16 ; go to block end
qq 21.9+49.15+35.21+20.27+53.33; if marks[sr] = 11 then
qq 51.15+49.21+35.27+35.33+10.39; begin trykvr; tryktekst(⟨<value_call⟩)
a14: arn p , hs b22 ; program point: printpoint(stack[sr])
gp 30c , hv a72 ; previous last used:= sr; go to a end
a5: hh a13 LB ; if marks[sr]=00
hs b16 ; then begin trykvr
qq 37.9+49.15+36.21+53.27+51.39; tryktekst(⟨<name_call⟩);
qq 49.9+35.15+35.21+10.27;
a13: hv a14 , hs b16 ; go to programpoint end
qq 39.9+41.15+38.21+51.27+59.33; if marks[sr] = 01 then begin trykvr
qq 51.15+49.21+35.27+35.33+10.39; tryktekst(⟨<proc._call⟩)
```

```

    gan r3 ; for i:= sr+1 step 1
    arn 30c , it 1 ; until previous last used -1 do
    ca (r 1) , hh r 3 ;
    arn p , sy 64 ; print(stack[i]) end
    hs a15 ;
    hv r-4 , hs b16 ; trykvr; tryktekst(⟨<retur⟩)
    qq 41.9+53.15+19.21+20.27+41.33+37.39;
    qq 10 ;
a9: hv a14 , hs b16 ; go to programpoint; block:
    qq 50.9+35.15+38.21+51.27+34.33+10.39; trykvr; tryktekst(⟨<block⟩)
a20: gp 28c , arn 29c ; j:= sr; b:= if j=
    ca p , hv a18 ; first point then go to
    arn p-1 , gp 33c ; variables; z:= j
    hv a19 , NB ; if marks[j-1] ≠ 01 then go to proceed; trykvr
    sy 64 , hs a15 ; print(stack[j-1])
    pp p-1 , hh a20 ; j:= j - 1; go to b; proceed: trukvr
a19: hs b16 ;
    qq 39.9+38.15+57.21+37.27+19.33+18.39; tryktekst(⟨<points⟩)
    qq 10 ;
a23: pp (29c) , arn p ; j:= first point
    hv a21 , LA ; c: if marks[j] = 10 then go to switch
    pp p 1 , hs b22 ; j:= j+1; printpoint(stack[j-1])
a24: arn 33c , nc p ;
    hh a23 , ; if j ≠ z then go to c
a18: arn(30c) , IPA ; variables:
    arn a18 , pp (30c) ; array mode:=
    arna18+1 , NPA ; marks[first var] = 00
    hh a25 , IPA ; j:= first var; go to P;
a21: pp (p) , hv a24 ; switch: j:= part 1(stack[j])
a28: hh a26 , NPA ; start: if -, array mode
    arn p , ; then go to noarr
    hv a27 , LA ; if marks[j] ≠ 00 then go to novar
    hs b16 , IPA ; array mode := false
    qq 21.9+49.15+41.21+57.27+49.33+50.39; trykvr
    qq 35.9+53.15+18.21+10.27 ; tryktekst(⟨<variables⟩);
a37: sy 64 , arn p ; printvar: trykvr
    hs a15 , ; print(stack[j])
a29: pp (30c) , pp p 1 ; next: j:= g; j:= j + 1
a25: gp 30c , arn 29c ; P: g:= j; if j≠ first point
    nc p , hv a28 ; then go to start

```

[7.9.65, Running system 18, track 17c61]

```

      pp (28c) , hv a1 ; go to follow chain
a27: hh a29 NB ; novar: if marks[j] = 10 then go to next
      hv a30 ; go to new declaration
a38: srnf32c , tkf -29 ; printarr:
      tk 30 , ga a33 ; k:= part 2(stack[j])
      arn p , ck 10 ;
a35: ga a22 , sy 64 ; line: trykvr
a36: pa a11 , can (a33); t:= 0
      hh a29 , ; d: if f = 0 then
      ps (a33) , ; go to next
a22: arn s , hs a15 ; print(stack[k-f])
a33: qq 0 , t 1 ; f:= f+1
a11: arn , D t 1 ; t:= t+1
      ca 5 , hh a35 ; if t=5 then go to line
      sy 0 , sy 0 ; else tryktekst(⟨<,,,,⟩);
      sy 0 , sy 0 ; go to d
a26: hh a36 , arn p ; noarr:
      hv a37 NA ; if marks[j] = 00 then go to printvar
a30: hs b16 IPA ; new decl: array mode:=true
      qq 49.9+41.15+41.21+49.27+24.33+10.39; trykvr; tryktekst(⟨<array⟩)
      pp p 1 , arn p ; seek: j:= j + 1
      hv r-1 LB ; if marks[j] = 11 then go to seek
      arn p 1 , gr 32c ; f:=stack[j+1]
      pp p 1 , sy 64 ; j:= j + 1; trykvr
a39: arn p , hs a15 ; dim: print(stack[j])
      arn 29c , ca p 1 ; if j+1= first point then
      pp (30c) , hv a38 ; h:begin j:= g; go to printarr end
      pp p 1 , arnfp ; j:= j + 1
      hv r-2 NA ; if marks[j] ≠ 10
      hv r-3 LB ; then go to h
      sy 0 , sy 60 ; tryktekst(⟨<_x⟩)
      sy 2 , sy 58 ;
      mkf 32c , grf 32c ; f:= f×stack[j]
      hv a39 ; go to dim
a15: gr b8 , gs a61 ; procedure print(x)
      arnf b8 , grf 31c ; real x
      arn b8 , sr 31c ; x in R-reg at
      hv a65 LZ ; start, if x not
      pa r1 t9 ; floating number only
      bt t-1 ; 9 spaces and sign=+- are
```

[8.9.65, Running system 19, track 18 c 61]

```

    sy    0    , hv  r-1    ;    printed
    arn  b8      ;
    sy   60      NT    ;
    sy   32    , sy   58    ;
a66: sy    0    , hv  b7+2    ;    exit
a65: snnf31c    , arf  b19    ; if abs(x)>228
    hv   a52    LT      ; then go to floating
    annf 31c    , tkf  -29    ; if entier(x)=x
    nkf   39    , snf  31c    ; then go to integer
    hh   a45    LZ      ;
a52: arnf31c    , ca    0    ; floating:
    sy    0    , hh   r 1    ; output sign
    sy   32    , sy   59    ; (+ or -) and point
a44: ps    0    , gs   b15    ; s:= 0, cycle:
    snnf31c    , arf  b20    ; if abs(x)<1then
    hv  a48      NT      ; go to mult
    snnf31c    , mkf  b18    ; x:=0.1xx
a51: ps  s1    , grf  31c    ; s:=s+1
    hh  a44      ,        ; go to cycle
a48: annf31c    , srf  b18    ; mult: if
    hv   a50      NT      ; abs(x)>0.1then
    annf31c    , dkf  b18    ; go to printing
    ps s-1      , hh  a51    ; x:=10xx; s:= s-1; go to cycle
a50: annf31c    , tkf   10    ; printing:
    ar   19c      IOB      ;
    arn  b17      LOB      ;
    qq  (b15)    LOB  1    ;
    pa  b 7  X   IRA  2    ; print
    pa  b 6      , hs   b 5    ; mantissa
    sy   60      , sy   27    ; print 10
    sy   58      , arn  b15    ;
    pa  b 6      IRAt  6    ;
    pa  b 7      t  2      ; print exponent
    tk  -30      , hs  3c    ; with sign
a45: hv  r4      , arnf 31c    ; print number
    tkf -29      , pm  b15    ; as integer
    pa  b 7      IRAt  8    ; with sign
    pa  b 6      , hs   3c    ;
a61: ps          , hv  a66    ;
a60: arn 27c      , ck  -30    ;
```

e ; end bloack RS 16 - 19

e ; end bloack RS 8 - 19

d c60=19c61 ;

s

[14.9.65]

[STANDARD PROCEDURES.

Enter a Standard procedure in the identifier list. example :

b k=c60+e70, i=0

Body of the standard procedure.

The local slipnames a1 and a2 are defined to:

a1 = track number of entrypoint (viz. k-e70)

a2 = track relative address of entry point

b k=e90, i=0

d i=e89

qq pass6 controlwordno.9 + a1.19 + a2.29 + pass6 output.39 pass6 marks

t name of standard-procedure;

d e71=e71+1; count number of standard identifiers

d e72=e72+L; count number of words used for long standard identifiers
in pass 2.

For a standard identifier of n characters (where caseshifts are
counted as characters) L may be calculated as:

L:= if n ≤ 6 then 0 else n:6;

This calculation will ensure enough space reservation for long words

d e89 = i;

e

d c60 = c60 + number of tracks used for the standard procedure

e ; end of the standard procedure

Control of the normal mode output from pass 6 through the control word:

f-marked words.

Part 4 followed by pars 3 and 2.

Not-f-marked words

Procedures with one parameter or no parameters: part 4.

Procedures with any number of parameters: d23=97 (CDC: 115)

followed by parts 3 and 2]

d e71 = 0 ; Number of std. identifiers

d e72 = 0 ; Number of words in supplementary

; table (long identifiers)

b k = c60 + e70, i = 10 ; begin block e0 - 16e0

d e0 = k - e70 ; e0 = first track for std- procs

[14.9.65]

[write, output, parameter handling and layout unpacking. track e0]

```
b a50 ; begin block. tracks e0-3e0

[write] pm c90 XV ; vy write
[output] pm ra X ; pm 1023
        ps (c46) , pm s1 ; s:=lastused; M:=layout description
        gr s1 X MRA ; stack medium
        pm (c47) DX 3 ;
        ga s1 , hs c22 ; stack param. counter:
        arn (c36) , ps (c46) ; layout value
a1: hv ra2 , it 1 ; , next parameter
        ps (c46) , pm c68 ; count last used
        arn s , ud s ; restore output sum
        gm 1023 , ud c89 ; restore medium
        pm s-1 IRC ; move return information
        gm s MRC ;
        pm c43 , gm c37 ; UV:=nonsense
        pm s1 , ud r-11 ; R:=layout; store medium and param. counter
a2: pm s2 , it 1 ; M:=next param. description; count param.
        bt (s1) , hv c51 ; . exit
        gr s2 , hs c22 ; store layout;
        ps (c46) , arn(c36) ; . parameter value
a : sr c43 , pm 1023 ; -nonsense
        gm c68 , ud s1 ; store output sum, select medium
        hh ral LZ ; if value=nonsens then goto next parameter
        arnf(c36) , it p ; value of parameter;
        pt c83 , pp 256 ; save p . minexp:= -256
        pa c85 X 509 ; numberpart:=true; M:=x
        ga c82 , arn s2 ; exp 2 , layout
        gr (c36) D ; store picture
        sr c36 , tk 14 ; unpack layout
        ga c84 , tk 10 ; b
a3h: ck -6 , ga c81 ; return to here for expprinting . h
        tk 10 , ck -18 ;
        gt c1 , tk 20 ; f1
        ck -6 , ga c80 ; d
        tk 11 ITA ; TA:=n=1
        ck -7 , gr c54 ; bEf2
        pp 0 V LZ ; if bE=0 then minexp:=0
        ca 1 , pp 10 ; if bE=1 then min exp:=-10
        ca 2 , pp 100 ; if bE=2 then min exp:=-100
        hs c2 , qq 1e0.29 ; transfer to next track
        qq ; unused
b k=e90, i=0 ;
d i=e89 ;
qq 193.9+e0.19+0.29 ; pass 6 description of write
t write; ; identifier
qq 193.9+e0.19+1.29 ; pass 6 description of output
t output; ; identifier
d e71=2e71, e89=i ; count identifiers
e ;
```

[14.9.65]

[conversion, exp 10. roundoff., track 1e0]

```

      ps  (c82)  t   10           ; s:=exp2 + 10
      ann   c53  ,   pa   c2       ; R:=abs(x1); expl0:=H:=0;
a6h:  pa   c47  ,   nk   r1       ; comment x1 is stored by c2 in c53
      ps   s0    ,   gr   c37      ; value to be printed=x=x1×2↑exp2;
      hv   ra12                LZ  ; if x1=0 then goto compute b
      pm   c37                                ;
      bs   s      t   -1          ; conversion:
      mkn  ra19  ,   hh   ra4      ; begin comment
      tk   s1    ,   gr   c37      ; by multiplication by
      ps   s7    ,   sr   ra18     ; 2↑3/10↑1 or 10↑1/2↑4
      mkn   320  DV              LT ; x is converted to form
a4h:  hv   ra7   ,   it   1        ; x=x2×10↑H where 1.0>x2≥.1
      qq  (c47)  t   -1          ;
a8:   ps   s-3   ,   hh   ra6      ; end conversion. x2 is stored in c37
a7:   arn  c84   ,   sr   c80      ; compute expl0:
      ga   ra10  ,   sr   c81      ; a10:=b-d; R:=b-d-h-1
      mb   c41   ,   it   (c81)    ; L : if H>h then
      bs  (c47)  ,   hs   ra14     ; begin Hh:=true; goto increxp10 end
      mt   ra8   ,   it   (c47)    ; R:=-R; L2: if H<b-d then
a10:  bs   0     ,   hs   ra15     ; begin Hh:=false; goto decrexp10 end
a11h: arn  ra10  ,   ar   c80      ; R:=b-d; L3: R:=R+d
a12:  ga   c82   ,   ps  (c82)    ; b1:=R;
      arn  256   D              NT ; rounding:
a13:  bs   s511  ,   hh   ra16     ; if b1 ≥0 then
      xr                ,   mkn  ra20 ; R:=.5×.1↑b1;
      ps   s-1   ,   hv   ra13     ; goto roundx2
a14:  bs  (c54)  ,   hv   ra15     ; increxp10: if bE>0 then goto a15
      bs  (c85)                ; if -,exppart then
      srn  (c54)  DV  1           ; begin R:=-1; bE:=1 end
      itn  (c47)  ,   pa   c81     ; else begin h:=H; R:=0 end
a15:  sc   c2    ,   bs  (c2)      ; decrexp10: expl0:=expl0+R
      ac   c47   ,   hh   s-1      ; if expl0-minexpl0>0 then
      ac   c2    ,   arn  c47      ; begin H:=H-R; goto if Hh then L else L2 end
a16h: hh   ra11  ,   ar   c37      ; expl0:=expl0-R; R:=H; goto L3; round x2:
      ps   -1                                ;
      hh   ra6                LO   ; R:=R+x2; if overflow then goto reconversion
      hs   c3    ,   qq   2e0.29 ; transfer to next track
a18:  vy   p51   ,   mln  (204)    ; comment constants .1-epsilon
a19:  can  s409  ,   cm   (r-410) ; 2↑3/10↑1
a20:  vy   p51   ,   mln  (s204)  ; .1+epsilon

```

[printing cycles, track 2e0]

```

hs   c65      ,   qq   3e0.29+1      ; call basic print
it   s7       ,   pa   ra26           ; update addresses for calls to
it   s4       ,   pa   ra29           ; basic print
it   s1       ,   pt   ra31           ;
gs   ra30     ,   ud   c1             ; p:=f1
bs   (c82)    ,   hv   ra21           ;
grn  c53      ,   arn  c80           ; begin x:=0; H:=
ca   0        ,   nt   (c85)         ;   if d=0^p>2^-,exppart then
bs   p-510    ,   it   1             ;   1 else 0;
pa   c47      ,   ar   c47           ;   b1:=d+H; exp10:=0; R:=0
ga   c82      ,   pan  c2           ; end
a21: pm  c47    X                   ; M:=R
bs   (c47)    ,                   ; if H>0 then
sc   c81      V                   ; begin h:=h-H; LT:=false end
qq   (c81)    t   -1             LTA ; else if LT then h:=h-1
bs   p509     ,   hv   ra25         ; if f1<3 then goto print leadingspaces
a22: arn (c85)  D   15             ; print ten and sign:
a23: hs   (ra30)      LT           ; if -,numberpart then print ten
pp   p10      ,   arn  c53         ; p:=p+10;      R:=x
arn  -480     DV             NT     ; plus
arn  32       DV             ; minus
bs   p500     ,   ck   10         ; if p<12 then space for plus
ca   p-10     ,   hh   ra24         ; if p=10 ^ x ≥ 0 then skip sign
a24: hs   (ra30) ,   pp   4         ; else print sign; sign printed:=true
a25: bt   (c81)  t   -1             ; print leading spaces:
a26: hsn [a46] ,   hv   ra25         ; count h
bs   p509     ,   hv   ra22         ; print sign if -, sign printed
a32: qq   59    ,   ps   r2         ;
hvn  (ra26)   t   -1             LTA ; if TA then print zero before point
bt   (c47)    t   -1             ; print digits before point:
a29: hsn [a44] ,   hv   r-1         ; count H
arn  ra32     ,   bs   (c80)         ; if d>0 then print point
a30: hs   [a41] ,   it   -1         ; print decimals
a31: bt   (c80) ,   hh   [a42]       ; count d
arn  c54      ,   ud   c83         ; R:=bEf2; restore p
pa   c82      V   -2             NZ ; if R=0 then
hs   c2       ,   qqf  e0.29+7     ; transfer to next parameter
ga   c84      ,   pm   c2         ; b:=bE; M:=exp10
hs   c2       ,   qqf  e0.29+29    ; transfer to expprint
qq                               ; unused

```


[14.9.65]

```

[basic print, sqrt, outsp, track 3e0]
a41: bs (c82) , hv ra45 ; if b1>0 output digit else
a42: hvn ra46 , arn ra50 ; output space
a43: it (c47) t 1 ; count H
      bs 0 , hvn ra41 ; if H<0 then output 0
a44: bt (c82) t -1 ; count b1
      mln ra50 , tk 30 ; next digit to R
a45: ar 16 D LZ ; Zero instead of space
a46: ga ra48 , bs (ra51) ; if actual case=upper then
      mt c41 , it 510 ; R:=-R;
a51: sy 570 t -510 LT ; if case changed then output case
      bsn p507 , arn c36 ; R:=if -,signprinting then picture else 0
      sy 0 LT ; if R<0 then output space
a48: sy 0 , ac c36 ; output; picture:=picture+R
      hhn s ; goback
a50: qq 10.39 ; 10
      qq ; unused
[sqrt]a47: ps (c46) , pm s1 ; call
      hs c22 ; parameter
      arnf(c36) , ps 8 [text] ; RF:= parameter;
      hh c55 LT ; if N < 0 then go to alarm
      grf c37 , ps (c46) ; x := N
      hv c51 LZ ; if N = 0 then exit
      arn c37 , gr c54 ;
      tk -1 ; exponent(x) :=
      ga c54 , pa ra49 ; entier(exponent(N)/2)
a33: arnf(c36) , dkf c54 ; for i := 1 step 1 until 5 do
      arf c54 X ; x := (N/x + x)/2
      sr 1 DX ;
a49: bt 0 t -128 ; i:= i + 1; if i>5 then begin
      grf c37 , hv c51 ; UV:= RF; exit end;
      grf c54 , hv ra33 ; c54:= RF; go to repeat
      qq ; unused
[outsp]
a34: ps (c46) , pm s1 ; take value
      hs c22 ; of parameter
      arnf(c36) , tkf -29 ; R:= round(RF);
      pm c43 , sr c34 ; M:= nonsense; Q: R:= R - 1;
      gm c37 V LT ; if R ≥ 0 then
      sy 0 , hh r-2 ; begin output space; go to Q end;
      ps (c46) , hv c51 ; UV:= M; exit
      qq ; unused
d a47=a47-a41, a34=a34-a41 ; relative addresses
b k=e90, i=0 ;
d i=e89 ;
qq 194.9+3e0.19+a47.29+92.39f ; pass 6 description of sqrt
t sqrt; ; identifier
qq 198.9+3e0.19+a34.29+92.39f ; pass 6 description of outsp
t outsp; ; identifier
d e71=2e71, e89=i ; count identifiers
e ;
e ; end use of a - a50 names

```

[14.9.65]

[integ exp, abs, entier, sign, outchar, writecr, outcr, track 4e0]

ba20

d e77=k-e70,e78=0 [track number and track relative for integerexp]

```

[integexp]a:arnf c68 ITA ; sign for exp
          annf c68 , tkf -29 ; exp to R39
          pm c40 , gm c53 ; result:=
a14: hv ra12 X NZ ; more bits in exp
          arnf c53 V NTA ; if exp < 0 then
          arnf c40 , dkf c53 ; result:= 1/result
          ps (c46) t 1 ; store exponent part; M:=number part
          ps s-1 , hv c2 ; exit
a12: cln -1 , gm c68 ; take last bit of exponent
          hh ra13 LZ ; last bit=0
          arnf c54 , mkf c53 ; result:=resultxargument
a13: grf c53 , arnfc54 ;
          mkf c54 , grf c54 ; argument:=argument ↑ 2
          arn c68 , hv ra14 ; next bit in exponent
[abs] a1: ps ra8 , hh ra4 ;
[entier]a2: ps ra5 , hh ra4 ;
[sign]a3: ps ra9 , hh ra4 ;
[outchar]a4:ps ra11 , is (c46) ;
          a8: pm s1 , hv c22 ;
          a10: annf (c36) , grf c37 ; abs
          a5: ps (c46) , hv c51 ; exit;
          arnf (c36) , srf c35 ; entier
          tkf -29 , nkf 39 ;
          a9: hh ra10 ;
          arnf (c36) ; sign
          arnf c41 V LT ;
          arnf c40 NZ ;
a11: hh ra10 ;
          arnf (c36) , tkf -29 ; outchar: R:= parameter
          ck -10 , ga r1 ;
          sy 0 , pm c43 ; punch char: M:= nonsense
          gm c37 , ca 63 ; UV:= M; if character = <TF> then
          ck -10 , sc 1023 ; outputsum:= outputsum - 63;
          hv ra5 ; goto exit;
          qq ; not used
[writecr]a6:pm 1023 , ud c90 ; select typewriter; M:= output sum;
          sy 64 , ud c89 ; write CAR RET; output sum:= M;
          gm 1023 ;
          arn c43 , hv c14 ; S: R:= nonsense; exit
[outcr] a7: sy 64 , hv r-1 ; punch CARRET; go to S
d a1=a1-a, a2=a2-a, a3=a3-a, a4=a4-a ; relative addresses
d a6=a6-a, a7=a7-a ; - -
b k=e90, i=0 ;
d i=e89 ;
qq199.9+4e0.19+a1.29+31.39 ; pass 6 description of abs
t abs; ; identifier
qq197.9+4e0.19+a2.29+33.39 ; pass 6 description of entier
t entier; ; identifier
qq197.9+4e0.19+a3.29+92.39 f ; pass 6 description of sign
t sign; ; identifier
qq198.9+4e0.19+a4.29+92.39 f ; pass 6 description of outchar
t outchar; ; identifier
qq191.9+4e0.19+a6.29+91.39 f ; pass 6 description of writecr
t writecr; ; identifier
qq191.9+4e0.19+a7.29+110.39 ; pass 6 description of outcr
t outcr; ; identifier
d e71=6e71, e72=2e72, e89=i ; count identifiers
e ;
e ; end use of a - a20 names
s

```

[14.9.65]

[Entry to algol from HP. Only relevant when placed on track 25
i.e. c60 defined to 1 on front page]

```
b a7, b2, d2          ; begin
d d1 = 80, d2 = d1 -40 ;   define buffer 1 and buffer 2;

b:  qq [e84-31c61-1]    ;   number of tracks to be summended and
[1b]qq [init transl track];   init transl track (set by TLA when e96 > 0);

a1: lk (rb1)   , pp 40    ; procedure sum; comment from track given by
vk (s)        t  1      ;   vk instruction in cell[s], number of
it (rb1)      , pt  rb2  ;   tracks -1 given as address in cell[s+1].
b1: nt  d1     , lk d1+d2 ;   Track 38 is skipped during summation:
is (s)        , it  s473 ;
bs -511      , hv  ra3   ;
b2: pp  p-1    , ar p-1   ;
bs  p         , hv  r-1   ;
a3: bt (s1)    t  -1     ;
hh  ral       ;
ck  0         , hh  s1   ;   exit;

a4: sy 64[CR] , sy 29[RED]; sum error or KC: writecr; write char(<RED>);
sy 60[UC] , vk  0      ;
hv  ra5      , LZ      ;   write text
sy 18[S]   , sy 20[U]  ;   (if R  $\neq$  0 then  $\{$ <SUM $\}$ 
sy 36[M]   v        ;
a5: sy 34[K] , sy 51[C] ;   else  $\{$ <KC $\}$ );
sy 0[SP]   , sy 49[A]  ;
sy 35[L]   , sy 55[G]  ;   write text ( $\{$ < ALGOL $\}$ );
sy 38[O]   , sy 35[L]  ;
sy 62[blk], lk  0     ;   write char(<BLACK>);
vy 17      ,         ;   by:= 17; comment remove HP-lock;
vk  0      , hv  33   ;   go to basic input on track 0;

a6: hv  ra4      NZ    ; test sum: if R  $\neq$  0  $\vee$  KC then
hv  ra4      LKC    ;   go to sum error or KC;
vk (r1b)   , lk  e25  ;   go to initialize translator on track e14;
vk (r1b)   , hv  e25  ;

d i=b+35          ;   define entry point from track 0;

vk 24c61   , lk  rb-40 ; from track 0: restore this track;
vk  0      , hsn ral-40; sum (track 0);
qq  0      , mt  r     ;   R:= -R;
vk 31c61   , hs  ral-40; sum (translator tracks from 31c61 and rest);
qq (rb-40), hv  ra6-40; go to test sum;

e                ; end Entry to algol from HP;

d e0=e0+e96, i=i+e99-40 ; This track is overwritten if e96 = 0
```

Standard procedures to drum and from drum.

```
integer procedure to drum(A);
array A;
```

```
integer procedure from drum(A);
array A;
```

Function:

Array A is written upon the drum by to drum and read from the drum by from drum. The standard integer drum place, which is the address of the first free location on the drum (calculated as drum place=trackno×40+trackrel). is assigned a new value.

The value of the procedures is the change of drum place, which is -length of array.

Successive calls of to drum will pack the arraya tightly upon the drum.

An alarm will be given if drum place takes on a value outside a restricted region. The restricted region is the entire drum, while reading from the drum, and the space between top of standardprocedures and the translated program, while writing upon the drum.

During writing upon the drum the hitherto smallest value of drum place is stored in c93 (MK: c92) as endtrack:2 with unit in position 9.

The code for to drum and from drum occupies two tracks and contains a normal blockentry, which reserves space for 6 local variables, which are named as follows:

Contents of the stack after blockentry:

p-6: working location	
p-5: endrel	corresponds to final value of drum place
p-4: endtrack	- - - - -
p-3: begrel	corresponds to initial value of drum place
p-2: begtrack	- - - - -
p-1: value of function	= -length of array
p : block information	
p+1: -	
p+2: -	
p+3: array identifier	
p+4: last used before call:	

Other variables are:

<u>Boolean</u> part	,LZA=true	
nospace	,LZB=true	
to drum	,NTA=true	
<u>integer</u> address	,contents of b8	current address in array
fixed place	,contents of b7	buffer address
lower limit	,contents of b11	

```

[12.10.65]
[track 5e0, to drum, from drum, track 1, lyn, char]
b b20 ;
d b10=6e0, e26=5e0 ; comment to drum is referenced by
[to drum] ; gierdrum as trackrelative 0;
b9: arn r V ITA ; to drum:= true; go to entry
[from drum] srn r ITA ; to drum:= false;
hs c5 , qq 1016.29+c1-1 ; entry: blockentry(6)locals
hh (c84) , ps 12 ; blocklevel:(1);
arnf c69 , tkf -29 ; M:=drum place
gr p-1 X IZC ; part:=nospace:=false;
dlr rb1 , tl 30 ;
gr p-2 , gm p-3 ; calculate begtrack and begrel;
arn p3 , ga rb4 ; R:=arrayidentifier;
arn p-1 ;
b4: sr 0 X -1 ITB ; M:= drumplace-length;
hh c55 LTB ; if drum place-length < 0 then alarm;
dlr rb1 , tl 30 ;
gr p-4 , gm p-5 ; calculate endtrack and endrel;
arn p-2 , ck -1 ; if to drum then
pm (40c67) DXV NTA ; begin
pm e97 DX ; if begtrack ≥ first track
gm p-1 , ck -1 ; v endtrack < e98
mt r-1 V NTA ; then alarm;
sr p-1 , hv rb13 ; if min drum place>endtrack:2 then
ar p-1 ; min drum place:=endtrack:2
hh c55 NT ; end
arn p-4 , ck -1 ; else
[23] qq [pm e98 DX] ; if begtrack>top drum
ck -1 , mt r ; then alarm;
gm p-1 , ar p-1 ;
hh c55 LT ;
it (p-1) , bs (c92) ;
arn p-1 , ga c92 ;
b13: hh c55 LT ;
srn (rb4) ;
nkf 39 , grf p-1 ; drum place:= drum place - length;
hs c2 , qq b10.29 ; goto track b10;
b1: qq 40.39 ;
qq ; unused
qq ; unused
[lyn] b14: lyn c68 V ; R:= input; go to P;
[char] b15: arn c49 ; R:= character
nkf 9 , hv c13 ; P: RF:= R; exit
qq ; not used
[word 23 is loaded by the translator loading administration]
d b14=b14-b9, b15=b15-b9 ; relative addresses
b k= e90, i=0 ;
d i=e89 ;
qq 147.9+5e0.19 +32.39f ; pass 6 word for to drum
t to drum; ; identifier
qq 147.9+5e0.19+ 1.29+32.39f ; pass 6 word for from drum
t from drum; ; identifier
qq 148.9+5e0.19+b14.29+111.39f ; pass 6 word for lyn
t lyn; ; identifier
qq 148.9+5e0.19+b15.29+ 91.39f ; pass 6 word for char
t char; ; identifier
qq 146.9+e80.29-e79.29+74.39f, ; pass 6 word for drum place
t drum place; ;
d e71=5e71, e72=2e72, e89=i ; count identifiers
e ;

```

[15.10.63]

[track 6e0, to drum, from drum, track 2]

```
vk (p-2) ;
arf c69 , grf c69 ;
ps (c46) , ps s-40 ; if last used-40 < topprogram
pmn s DX ; then begin fixed place:=c67;
ck -1 , sr c93 ; nospace:=true end else
psn c67 LT ; fixed place:=last used-40;
gs rb7 IZB ;
arn p3 , tk 10 ; address:=part2(arrayidentifier)
ga rb8 , arn p-3 ; if begrel=39 then
ca 39 , hv rb6 ; goto fulltracks;
arn p-2 , ca (p-4) ; if begtrack=endtrack then
hv rb5 ; goto last track;
pt rb11 , hs rb ; lower limit:=0; move first part;
pa p-3 t 39 ; begrel:=39;
b3: vk (p-2) t -1 ; counting: begtrack:=begtrack-1;
b6: ps 0 , arn p-2 ; fulltracks: if begtrack=endtrack
ca (p-4) , hv rb5 ; then goto last track;
sk (rb8) V -40 NTA ; address:=address-40;
lk (rb8) t -40 ; if to drum then write else read;
b5h: hv rb3 , ps (p-5) ;
it s1 , pt rb11 ; lower limit:=endrel+1;
bs s473 , hs rb ; if endrel < 39 then move last part;
vk c64 , ps p4 ; if nospace^part then
lk c67 LZC ; restore running system;
vk c64 , hh c7 ; goto exit function;
b: gs rb2 , ps 0 ; procedure move;
lk (rb7) , arn rb8 ; read track to fixed place;
sr 1 D ;
sr p-3 , ga rb8 ; for s:=begrel step 1 until
ps (p-3) , vk (p-2) ; lower limit do
b12: qq V LTA ; if to drum then
b8: pm s[address] V IRC ; STACK[fixed place+s]:=STACK[address+s]
b7: pm s[fixed place] VIRC ; else
gm (rb7) V MRC ; STACK[address+s]:=STACK[fixed place+s];
gm (rb8) MRC ;
b11: ps s-1 , it 0 ;
bs s1 , hv rb12 ;
psn 0 IZA ; part:=true;
sk (rb7) NTA ; if to drum then write from fixed place;
b2: ps 0 , hv s1 ; return;
e ; end use of b - b20
```

[15.10.63]

[outtext, writetext, track 7e0]

```
[outtext]    arn  r9      ,  hh  r1      ; R:= , pm 1023
[write text] arn  c90     ,  tk  10      ; R:= , vy write
ck   -10     ,  ps  (c46)    ;
ar   (c47)   D  1          ; R:=R+ appetite+1
pm   c43     ,  gm  c37     ; UV:=nonsense
pm   s1      ,  ca  0       ; M:=parameter
vk   0       ,  hv  c15     ; exit if no more parameters
gr   s1      ,           MRA ; stack parameter counting and medium
vk   0       ,  hs  c22     ; get parameter
ps   (c46)   ,  pm  1023    ;
gm   c68     ,  ud  s1      ; store output sum select medium
pmn  (c36)   X           ITA ; value of parameter
gt   r5      V           NT  ; long text on drum or nonsense
cl   34      ,  hv  r11     ; short text
ck   -10     ,  ga  r5      ;
ck   10      ,  sr  c43     ;
hv   r17     ,           LZ  ; if parameter = nonsense then go to next
vk   (r2)    ,  ps  0       ;
lk   c67     ,  ps  s-40    ; get text from drum
vk   0       t  -1         ;
bs   s1      ,  hv  r-2     ;
arn  s40c67  ,  cl  36      ; next word
ar   32      D           LA  ;
ar   16      D           LB  ;
pa   r5      ,  ck  -4      ; character counter :=0;
ga   r3      ,  ca  10      ; next character
vk   c64     ,  hh  r5      ; finished
ca   63     ,  it  1       ; test for CR
sy   0       ,  cln -6      ; output character
bt   0       t  -80        ; count characters
ps   s1      ,  hv  r-10    ; next word
hh   r-7     ,  ps  (c46)   ; next character
lk   c67     ,           NTA ; restore free track place if used
pm   c68     ,  ud  s1      ; restore output sum
gm   1023    ,  ud  c89     ; restore by
arn  s1      ,  pa  c47     ; R:=parameter counting and medium
pm   (c46)   ,           IRC ; move
gm   (c46)   t  1          MRC ; return information
hh   r-36    ,           ; goto test for more parameters
qq                                     ; unused
;
b k=e90,  i=0
d i=e89
qq193.9+7e0.19+  0.29    ; pass 6 word for outtext
; identifier
t outtext;          qq193.9+7e0.19+  1.29    ; pass 6 word for writetext
; identifier
t writetext;
d e71=2e71,  e72=2e72,  e89=i
; count identifiers
e
;
```

```

[exp, writechar, track 8e0]
b a14 ;
d e75 = k-e70, e76=0 ; [track no., track relative for exp]
[exp]
a12: ps (c46) , pm s1 ;
hs c22 ;
annf(c36) , srf ra0 ; R:= abs(x) - 511 xln(2)
hh ral NT ; if abs(x) ≥ 511 × ln(2) then go to large
arnf (c36)x ; take argument
ga ra2 , mln ra3 ;
a2: tl 0 t 3 ; R9M := x/ln(2)
ga ra4 , tl 10 ; y1 := entier(R9M)
cl -2 ;
sr 128 DX ; M := R - 1/4
gm c37 , mkn c37 ; z := M; R := z↑2
pm 384 DX ; R := 0.75; M := z↑2
mk ra6 , gr c54 ; p := 3/4×(1 + p × z↑2)
gr c68 , arn ra7 ;
mk ra8 X ;
mkn c37 , sc c68 ; q := p - 3/4×(axz + c × z↑3)
ar c54 X ;
mln ra5 , dl c68 ; R := p/q/sqrt(2)
a4: nkf 0 t 1 ; RF := R×2↑(y1+1)
a9: grf c37 , ps (c46) ; UV := exp(x)
a1: hv c51 , arfn(c36) ; return; large:
hvn ral0 X LT ; if R<0 then begin exp:=0; go to exit end;
ps 9 , hh c55 ; if R>0 then go to alarm
f ;
a0: 354.1982 ; 511 × ln(2)
m ;
a3: qq 5.3+12.7+5.11+5.15+1.19+13.23+9.27+4.31+10.35+14.39 ; log base2(e)/2
a7: 0.519 860 384 45 ; 3/4xa
a6: 0.144 099 511 28 ; 3/4xb
a8: 0.016 626 103 230 ; 3/4xc
a5: 0.707 106 781 1865 ; sqrt(2)/2
qq ; not used
qq ; not used
[writechar]
a11: ps (c46) , pm s1 ; call
hs c22 ; parameter
arnf(c36) , tkf -29 ; R:= round(parameter)
pm 1023 , ud c90 ; save output sum; select typewriter
ck -10 , ga r1 ;
sy 0 , ud c89 ; type char; select punch
gm 1023 , pm c43 ; restore output sum; M:= nonsense
a10: gm c37 ; set UV: UV:= M;
ps (c46) , hv c51 ; exit
d a11=a11-a12 ; relative address
b k=e90, i=0 ;
d i=e89 ;
qq 194.9+e75.19+92.39f ; pass 6 word for exp
t exp; ; identifier
qq 198.9+8e0.19+a11.29+92.39f ; pass 6 word for writechar
t writechar; ; identifier
d e71=2e71, e72=1e72, e89=i ; count identifiers
e ;
e ; end a - a14 names

```


[15.10.63]

[outcopy, writecopy, input track 1, 9e0]

```
[outcopy]  ps  (c46)  ,  pm  s1                ;
           hs   c22                ; get parameter;
           arn  r      ,  hv  r7          ; tryk bool:=true;
[writecopy] ps  (c46)  ,  pm  s1                ;
           hs   c22                ; get parameter
           pm  1023  ,  ud  c90          ; save output sum; select write
           gm   c68  ,  arn c89          ;
           ck   10   ,  ga  r1          ; select
           vyn  0     t  1016          ;   input from tape (input)
           gt   r18                ;
           pm  (c36)  ,  pan c47          ; parameter      ; param case:=0
           hs   r21  ,  ga  r6          ; test character 1 is set
           hs   r20  ,  ga  r1          ; test character 2 is set
           pp   0     ,  pa  c47          ; p:=testcharacter 2; output case:=0
           hs   c65  ,  qq  12e0.29+1    ; call basic read
[next]     hsn (c91)                IOA    ; basic read
           ab   c48  ,  ga  c49          ;
           ca   0     ,  hv  r7          ; if R  $\neq$  testcharacter 1 then
           ncn  p-10 ,  hvn r-3          ; begin if testcharacter 2  $\neq$  10 then
           arn  c48  ,  ca  (c47)        ;   goto next
           sy   (c49) ,  hv  r-5          ;   else output character
           sy   60    V              NZ   ;   including case
           sy   58                ; end;
           ga   c47  ,  hv  r-3          ;
           gp   r-7  ,  ncn p-10          ; testcharacter 1:=p; if p  $\neq$  10 then
           pp   10   ,  hv  r-10         ; begin p:=10; go to next end;
           bs   (c48) ,  sy  58          ; output must end in LC;
           pm   c68  ,  can 0           ; if -, tryk bool then
           gm   1023 ,  ud  c89          ; restore sum select læs
           pm   c43  ,  gm  c37          ; UV:=nonsense
           ps   (c46) ,  hv  c51          ; exit
           it   128  ,  pa  c47          ; subroutine for unpack parameter
           cln  -6   ,  tk  -4           ;
           ca   58   ,  hh  r-2          ; test for LC
           ca   60   ,  hv  r-3          ; test for UC
           ca   63                ; test for CR
           ar   1     D                ;
           ar   c47  ,  hh  s            ;
           qq                ; not used
           qq                ; not used
b k=e90,  i=0                ;
d i=e89                ;
           qq 195.9+9e0.19  + 93.39f    ; pass 6 word for outcopy
t outcopy;                ; identifier
           qq 195.9+9e0.19+3.29+93.39f  ; pass 6 word for writecopy
t writecopy;              ; identifier
d e71=2e71,  e72=2e72,  e89=i          ; count identifiers
e
```

[14.9.65]

[input, typein, inone, typechar, inchar, input track 2, 10e0]

```
b a10
[input] a: srn (c47) DX 1 ; appetite+1
ps (c46) , hv r6 ; s:=lastused
ps (c46) t 1 ; s:=lastused:=lastused+1
pm s-1 IRC ; return information
arn (s) D -1 IZA ; count parameters
gm s X MRC ; move return information
hv c15 LZA ; exit if no more parameters
arn s1 , gm s1 ; next parameter, save counter
hh r18 X LA ; expression
ga c36 , tk 10 ;
ca 0 , hvn r4 ; simple variable
ar (c36) t -1 ; array length
ga c36 , tk 30 ;
sc c36 , sr r-3 ;
pm r22 , ga c47 ; M:=return 1; set number of numbers
hs c65 , qq 11e0.29+1 ; call number read
gm s31 , hv r21 ; set return; goto number read
[typein] a1:pm r17 ; M:=return 2
[typechar]
a2: hh r9 ; (s=r-1 or r-2) save sum and case
[inone] a3: pm r16 , ud c12 ; M:=return 3
hhn r-6 , it r11 ;
[inchar]a4: pa r5 t c12 ; s&t return from in-or typechar
hs c65 , qq 12e0.29+1 ; call basic read
hsn (c91) IOA ; basic read
ab c48 , ga c49 ;
nkf 9 , grf c37 ; set UV
hv 0 , hs c22 ; exit or restore sum and case
hvn r-13 , arn c63 ; expression in input; save
gr c68 , ud c90 ; sum and case
it (c48) , pa c82 ; select type
pa c48 , hhn s3 ; case:=0; return from save
pm c68 , ud c89 ; restore sum and case
it (c82) , pa c48 ; select input
gm c63 , hv c12 ; exit
hs c2 , qq 10e0.29+31 ; exit from number read after typein
hv c12 ; , , , , , inone
hs c2 , qq 10e0.29+2 ; , , , , , input
qq (c47) t 1 ; n:= n + 1
gp s29 , hv s1 ; save p, go to number read
qq ; not used
d a1=a1-a, a2=a2-a, a3=a3-a, a4=a4-a ; relative addresses
b k=e90, i=0 ;
d i=e89 ;
qq 192.9+10e0.19 ; pass 6 word for input
t input; ; identifier
qq 190.9+10e0.19+a1.29+ 91.39 f ; pass 6 word for typein
t typein; ; identifier
qq 148.9+10e0.19+a2.29+ 91.39 f ; pass 6 word for typechar
t typechar; ; identifier
qq 190.9+10e0.19+a3.29+ 91.39 f ; pass 6 word for inone
t inone; ; identifier
qq 148.9+10e0.19+a4.29+ 91.39 f ; pass 6 word for in char
t in char; ; identifier
d e71=5e71, e72=1e72, e89=i ; count identifiers
e ;
e ; end a - a10 names
```

[14.9.65]

```

[numberread, input track 3, 11e0]
b b10 ; begin block for 11e0 to 14 e0
b a20 ;
a8: qq (c36) V 1 ; UA:=UA+1;
[entry] a3: hs c65 , qq 12e0.29+1 ; call basic read
ppn 0 , ud ra1 ; state:=0; neg:=false;expfac:=10
pm c40 , gm c53 ; dec factor:=1
a5: grn c37 , grn c54 ; number:=0; exp:=0
[next] a: hsn (c91) IZA ; basic read
qq 15.6+ 6.12+436.21+272.30+ 5.39; terminator parameter
qq 31.6+26.12+ 65.21+ 31.30+ 71.39; digit ,
qq 27.6+14.12+511.21+464.30+511.39; minus ,
qq 0.6+23.12+511.21+471.30+511.39; plus ,
qq 0.6+35.12+143.21+511.30+511.39; point ,
qq 7.6+53.12+287.21+127.30+511.39; ten ,
[ten] it (s29) , bs 9 ; if oldstate < 3 then
pm c40 , gm c37 ; number:=1;
[digit h]a13: hv ra , grf c54 ; goto next
arnf c37 , mkf ra2 ; number:=number×10+digit;
arf c54 , grf c37 ;
arnf c53 , bs p-3 ; if state > 3 then
a7: mkf ra2 , grf c53 ; dec factor:=dec factor×10
[ditto h] hh ra5 , hh ra6 ; goto next; ;
[termin] gp r1 , it ra ;
hh p0 , hh ra4 ; goto terminator action [f(state)]
[.1] a11: cm (r-4) , can r409 ;
[10.0] a2: qq 3 , qq 320 ;
[end number] arnf c37 , dkf c53 ; store [UA]:=number /
bt (c54) t -1 ; decfactor × expfactor ↑ exp
a9: mkf r0 , hv r-1 ;
mkf c41 NZB ; if neg then number:=-number
a6: grf (c36) , ud c89 ;
pp 0 , it -1 ; restore p ;
ncn (c47) , hv ra8 ; go back if more numbers;
qq 0 , qq 0 ; exit instruction set by number read
[minus] bs p-1 ; minus
pa ra9 V a10 ; set exponent factor
a1: pa ra9 t a12 IZB ; - - - ; neg:=true
[alarm h] a4: hv ra , hsn sb2 ;
[digit] arn c49 , bs p506 ; digit
nkf 9 , hh ra13 ; not in exponent
pm c54 , ml sb3 ; in exponent
gm c54 , hv ra ; exp:=exp × 10 + digit; goto next
d b5=a3-a4 ; return after error, relative
d a10=a11-a9, a12=a2-a9 ;
e ; end a to a20 names

```

[14.9.65]

[basic read, input track 4, 12e0]

b a10

```

a1: it      128      , pa  c48      ; set case
[entry] a: lyn  c49    V              ; normal entry: P:= input
      qq              V              ;
      qq              ;
      qq              ;
      ca      63      , hv  ra      ; ignore TAPE FEED
      ca     127      , hv  ra      ; - ALL HOLES
      ca     31      , hv  ra2     ; punchoff
      ck    -10      , ac  c63     ; sum:=sum+character
      ck     10      , ca  28      ;
      grn   c63      , hv  ra      ; clear code
      ca     60      , hv  ra1     ; UPPER CASE
      ca     58      , hh  ra1     ; LOWER CASE
      ca     12      , hv  ra3     ; end code
      ca     61      , hv  ra3     ; sum code
      ar     c48      ,          NZ ; if not space then R:= R + case
a7: hr      s1              NZA    ; return to outcopy or inchar
      hv     ra              LZ    ; if space then go to read HT: arn (c49) D
      ca     14      , hv  ra      ; underline
      ca     16      , qqn         ; zero
      ga     c49      , pm  s1      ; set last character
      it    (c49)     , bs  10      ;
      pm     s2      , hv  ra4     ; digit
      ca     32      , pm  s3      ; minus
      ca    160      , pm  s4      ; plus
      ca     59      , pm  s5      ; point
      ca    155      , pm  s6      ; ten
a4: tln     26      , gt  ra5      ;
      gp     ra6      , tl  p20     ; p contains state for number read
a6: ck      p0       , tk  -7      ;
      ga     ra5      , nc  7      ; state increment
a5: pp      p0       , hr  s0      ; ok, exit to number read
      pp     10      , hv  ra      ; forbidden character in number read
a3: vk     13e0      , lk  c67     ; call error track
      vk     13e0      , hv  c67
[special entry]
a9: pa     c91      t  1          ; char is set:= false
      arn    c49      , hv  ra7     ; use last character
a8: ca     44      , hv  ra      ; punch on
a2: lyn    c49      , hv  ra8     ; reading after punch off
a10: qq    10.39      ; constant
d b1=a9-a1      ; special entry, relative
d b2=a3-a1      ; error entry, relative
d b3=a10-a1     ; constant 10, relative
e              ; end a - a10 names
```

[14.9.65]

[error- and pause- output during read]
[input track 5, 13e0]

b a20

```
[entry] hv    ra          LZ      ; error from numberread
        gk    ra1    ,   ca    61      ;
        arn   c63    ,   hh    ra2      ; sum code
        hs    ra3          ; text
        qq    58.9+39.15+49.21+20.27+18.33+53.39; LC p a u s e
a2: hv    ra4    ,   tl    -59      ;      , test inputsum
        dl    ra5    ,   cl    -10      ;
        ga    ra6    ,   lyn   c49      ;
        pa    c63    ,   pt    c63      ;
a6: ca    _0      ,   hh    ra4      ; sum ok
        hs    ra3          ; text
        qq    29.9+18.15+20.21+36.27+54.39; rødt S u m _ f
        qq    49.9+57.15+35.21+18.27+62.33; a i l s black
a4: lyn   ra6    ,   gs    ra7      ; wait for input to go on from
a1: vy    _0      ,   qq          ; pause and sum , restore by
        hr    ra8          ; get start address for basic read
a8: it    sl      ,   pt    ra10      ; set return address
a7: ps    _0      ,   hv    ra9      ; restore s , exit
a3: pm    1023    ,   vy    17      ; subroutine for print text;
        syn   64    ,   gs    ra11     ;
a11: arn   _0      t    1          LZ   ;
        hh    ra12          LA      ;
        ga    ra13    ,   tk    10      ;
a13: sy    _0      ,   ck    -4      ;
a12: hv    ra11    ,   sy    58      ;
        gm    1023    ,   hr   (ra11)   ;
a5: qq    127.39          ; constant for sumcheck
        qq          ; not used
a: bs    (c48)    ,   it    20      ; set U or L in number error text
        pa    ra14    t    35      ;
        hs    ra3          ; text
        qq    29.9+51.15+38.21+41.27+41.33+53.39 ; red c o r r e
        qq    51.9+19.15+          57.27+37.33+39.39 ; c t _ i n p
        qq    20.9+19.15+          21.27+49.33+35.39 ; u t _ v a l
        qq    20.9+53.15+27.21+          53.33+37.39 ; u e , _ e n
        qq    52.9+          57.21+37.27+          60.39 ; d _ i n _ UC
a14: qq    _0.9 +51.15+59.21+          62.33+58.39 ; _ C : _ black LC
        it    sb5    ,   pt    ra10      ; set return addres for number error
        pa    c48          ; inputcase lower
a9: vk    c64    ,   lk    c67      ; restore free track place
a10:[vk    c64    ,   hv    _0      instructions in running system]
```

e ; end a to a20 names

[15.10.53]

[cos, sin, setchar, track 14e0, algorithm as described in TRIG-2, with one term less, see also DASK ALGOL]

ba10

[cos]

```
  a: ps  (c46) ,  pm  s1      ; call
      hs  c22                ;   parameter
      hvn  ra3                IZA ; cos:= true; go to Z;
```

[sin]

```
  a1: ps  (c46) ,  pm  s1      ; call
      hs  c22                ;   parameter
      arn  r                  IZA ; cos:=false;
  a3: arnf(c36) X              ; Z: RF:=parameter
      ga  r1 ,  ps  (c46)      ;
      bs  Xt -16              ; if RF < 2↑(-15)  $\wedge$  RF  $\neq$  0 then
      dkf  ra4 ,  hv  ra5      ; begin
  a6: arnf c40                  ; small: if cos then RF:= 1
  a8: grf  c37 ,  hv  c51      ; out: UV:= RF; exit end;
  a5: hv  ra6                  LZ ; if RF = 0 then go to small;
      tkf  10                  ; R:=modulus(F);
      ar  128 D                LZA ; if -, cos then R:= R + 1/4;
      tk  2 ,  gr  c53         ;
      pm  c53 ,  pt  ra7      ;
      pt  ra7 t -3 NO         ;
      srn 256 D                ;
      mk  c53 ,  gr  c54      ;
      pan  r1 t                all ;
  a10: ar  r4 Xt                1 ;
      arn  c53 V                LA ;
      mkn  c54 ,  hv  r-2      ;
  a7: mk  c53 ,  mt  r0        ;
      nkf  0 ,  hv  ra8      ; RF:= R; go to out;
```

m

```
  a9: -.000 003 418 229      ;
      +.000 151 656 333      ;
      -.004 369 731 387      ;
      +.072 906 209 920      ;
      -.569 703 680 149      ;
      +.267 162 131 344a     ;
```

f

```
  a4: qq 2.9+25.15+33.23+251.31+84.39 ; 2 $\times$ pi
```

m

```
d a11=a9-a10-1              ;
      qq                      ; not used
```

[set char]

```
  a2: ps  (c46),  pm  s1      ; call
      hs  c22                ; parameter
      arnf(c36) ,  it  b1      ; set special entry into
      pa  c91 ,  tkf  1        ;   track 12e0
      ga  c49 ,  ps  (c46)      ; char:= parameter
      hv  c51                  ; exit
```

```
d a1=a1-a,  a2=a2-a          ; relative addresses
```

```
b k=e90,  i=0                ;
```

```
d i=e89                      ;
```

```
qq194.9+14e0.19+92.39f      ; pass 6 word for cos
```

```
t cos;                       ; identifier
```

```
qq194.9+14e0.19+a1.29+92.39f ; pass 6 word for sin
```

```
t sin;                       ; identifier
```

```
qq149.9+14e0.19+a2.29+92.39f ; pass 6 word for set char
```

```
t setchar ;                  ; identifier
```

```
d e71=3e71,  e72=1e72,  e89=i ; count identifiers
```

e

e ; end a to a10 names

e ; end b to b10 names (track 11e0 to 14e0)

[14.9.65]

[arctan, track 15e0, the method is that described by Lance in Numerical Methods, page 40, supplemented close to the origin by the two first terms of a modified Taylor expansion]

```
b a12
[arctan]
  ps (c46) , pm s1 ; entry point
  hs c22 ; call arg
  arnf(c36) X IZA ; take arg
  ga ra2 XV LT ; if exponent ≥ 0 then
  arnf c41 , hh ral ; begin RF:= -1; go to big argument end
a2: bs [exp] t -8 ; if exponent > -8 then
  ps a10 , hh ra3 ; begin base:= pi/16; go to compute end
  mkf(c36) , mkf(c36) ; RF:= -x3×0.333306 + x;
  mkf ra4 , arf(c36) ; go to exit;
a1: hh ra5 , ps a11 ; big argument: base:= -3×pi/16;
  hhn ra5 LZA ; if arg = 0 then begin RF:= 0; go to exit end
a3: dkf(c36) , ga ra2 ; arg:= -1/arg; compute:
  tkf 9 , gs ra6 ; a2:= sign(arg);
  gr c68 , snn c68 ; c68:= arg/2;
  pm 128 DX ; c37:=
  mk ra7 , gr c37 ; .25 + (sqrt(2) - 1)/4×abs(arg);
  sr 256 D ;
  an c68 , tk -1 ; c37:=
  dk c37 ; (c37 - .5 + abs(arg)/2)/2/c37;
  gr c37 X ; c54:=
  mkn c37 , gr c54 ; c372;
  pan r1 t 5 ;
  ar r0 Xt 1 ;
  mkn c37 V LA ; R:= 37× polynomium;
  mkn c54 , hv r-2 ;
a6: arr[base], mt ra2 ; R:= (base + R)×sign(a2);
a5: nkf 1 , grf c37 ; RF:= 2×R; exit: UV:= RF;
  ps (c46) , hv c51 ; go to administration
  qq 132 189.26; a[13]
  qq -1 416 698.28; a[11]
  qq 10 874 463.30; a[9]
  qq -82 115 957.32; a[7]
  qq 670 327 015.34; a[5]
  qq -6 511 656 473.36; a[3]
  qq 113858 157095.38; a[1]
a7: qq-26.7-130.15-121.23-153.31-253.39; -(sqrt(2) - 1)/2
a8: qq-75.7- 99.15-753.23-252.31-205.39; -3×pi/16
a9: qq 25.7+ 33.15+251.23+ 84.31+ 68.39; pi/16
f ;
a4: -0.333 306 ; modified Taylor coeff
m ;
qq ; not used
d a10=a9-a6, a11=a8-a6 ; relative addresses
b k=e90, i=0 ;
d i =e89 ;
qq 194.9+15e0.19+92.39 f ; pass 6 word for arctan
t arctan; ; identifier
d e71=1e71, e89=i ; count identifiers
e ;
e ; end a to a12 names
s
```

[14.9.65]

[ln, outclear, outsum, track 16e0]

b a20

d e73=k-e70, e74=0 [track no., track relative for ln]

[ln]

```
a: ps (c46) , pm s1 ;
hs c22 ;
arnf(c36) , ps 5 ; RF:=arg; set error text
hh c55 LT ; if x<0 then go to alarm;
tk 9 VX NZ ; if x=0 then
pm ra1 , hv ra3 ; begin ln := -2↑166; go to exit end;
ga ra4 , gm c53 ; exp2 := exponent(x) × 2↑(-9) + 2↑(-10);
gm c54 , arn ra5 ; c53 := (sqrt(2)/4 - numerus(x)/4)/
ac c53 , sr c54 ; (sqrt(2)/4 + numerus(x)/4)
dk c53 , gr c53 ;
pm c53 , mkn c53 ; c54 := c53↑2;
gr c54 ;
pan r1 t a6 ; R := 0;
a8:ar r0 X 1 ; for k := -0.000 847 09, -0.001 126 143 206,
mkn c53 V LA ; -0.001 878 518 085 do
mkn c54 , hv r-2 ; R := (k + R) × c54;
ar ra4 X ; ln := ((R - .005 635 527 485) × c53 + exp2)
mkn ra9 ; × ln(2)
nkf 9 , grf c37 ; × 2↑9;
a2:ps (c46) , hv c51 ; go to exit
m ;
a7:-0.000 847 09 ; a[7]
-0.001 126 143 206 ; a[5]
-0.001 878 518 085 ; a[3]
-0.005 635 527 485a ; a[1]
a9:qq 354.9+57.15+5.22+126.29+31.34+9.39 ; ln(2)
a1: qq 165.9+1.10 ; -2↑166
a5:qq 5.4+10.8+8.12+2.16+7.20+9.24+9.28+9.32+15.36+6.39 ; sqrt(2)/4
a4:qq 0 t 512 ; exp2
a3:gm c37 , hv ra2 ; UV:= M; go to exit;
qq ; not used
qq ; not used
qq ; not used
[outclear]
a10:sy 28 , hv ra11 ; punch CLEAR CODE; go to w
[outsum]
a12:sy 11 , sy 61 ; punch STOP CODE; punch SUM CODE;
arn 1023 , tl -59 ; form sum
dl ra13 , cl -10 ; character
ga c47 , sy (c47) ; punch sum character
a11:pa 1023 , pt 1023 ; w: output sum:= 0;
arn c43 , hv c14 ; R:= nonsense; exit
a13:qq 127.39 ; modulus
d a6=a7-a8-1 ;
d a10=a10-a, a12=a12-a ; relative addresses
b k=e90, i=0 ;
d i=e89 ;
qq194.9+e73.19+92.39 f ; pass 6 word for ln
t ln ; identifier
qq 191.9+16e0.19+a10.29+91.39 f ; pass 6 word for outclear
t outclear; ; identifier
qq191.9+16e0.19+a12.29+91.39 f ; pass 6 word for outsum
t outsum; ; identifier
d e71=3e71, e72=1e72, e89=i ; count identifiers
e ;
e ; end a to a20 names
d c60=1e73+e96, e0=e0-e96 ; set running track and redefine e0
e ; end drum block, e0 - 16e0
```


[14.9.65]

[split and pack, page 2]

b k=c60+e70, i=10, a10

```
a2: arn ra4 , hh ra1 ; split: shift:= -39; go to common;
a1: arn ra6 , ps (c46) ; pack: shift:= 1;
    pm (c47) D 4 ; common:
    gm s-4 , gr s-3 ; n:= appetite+1;
    hs c5 , qq 1018.29+c1-1; blockentry (4) locals blocklevel: (1);
    hh (c84) , arn p3 ;
    pm p3 DX ; i:= p+3;
    gr p-3 , hs c22 ; take formal(store[i]);
    arn c36 , ca c37 ; if UA:= location(UV) then
    pm c37 , arn p-3 ; begin a:= p+3; store[p+3]:= UV end
    gr p-4 , gm p3 ; else a:= UA;
    arn p-2 , hv ra9 ; R:= n; go to test;
                                ; next parameter group:
a4: qq -39 , hs c22 ; take formal(store[i]);
    arnf(c36) , tkf -29 ; b:= address(store[UA]);
    ck -10 , ud ra7 ; i:= i+1;
    gr p4 , hs c22 ; take formal(store[i]);
    arnf(c36) , tkf -29 ; c:= address(store[UA])+ shift-b;
    ck -10 , ar p-1 ;
    sr p4 , ud ra7 ;
    gr p5 , hs c22 ; i:= i+1;
    arnf(c36) , tkf -29 ; take formal(store[i]);
    gr p6 , pmn(p-4) ; d:= round(store[UA]);
    ar 256 D LA ;
    ar 128 D LB ; RM:=store42bits[a];
a6: xr 1 , cl (p4) ; cl(b); tl(c);
    tl (p5) , bs (p-1) ; if shift ≤ 0 then
    cl -40 , hv ra5 ; begin
    nkf 39 , grf (c36) ; store[UA]:= float(R); M:= store[UA];
    pm (c36) , hv ra3 ; end
a5: ar p6 , it (p4) ; else
    ns (p5) , cl s40 ; begin cl(-40);
a8: xr -1 , ck 1 ; R:= R+d; cl(40-b-c);
    ga ra8 , pi (ra8) ; store42bits[a]:= RM
    gm (p-4) t MOC ; end;
a3: arn(p-2) D 3 ; n:= n+3; i:= i+1;
a9: ps (p-3) V 1 NT ; test: if n 0 then go to next parameter group;
a7: pm (p-3) t 1 ; s:= i;
    hh ra4 , LT ; if n=2 then
    ca 2 , hv c6-1 ; exit from function with value in M;
    ps 15 , hh c55 ; alarm(⌊<param⌋);
```

b k=e90, i=0 ; prepare loading of pass 6 words and

d a1=a1-a2, a2=0 ; procedure names

d i=e89 ;

qq145.9+c60.19+a1.29; pass 6 word for pack

t pack; ; name of pack

qq143.9+c60.19+a2.29; pass 6 word of split

t split; ; name of split

d e71=2e71, e89=i ;

e ;

d c60=1c60 ; increase running track

e ; end a to a10 names

[14.9.65]

[gier drum, gier proc and gier, page 1. kb on.

The code contains a normal blockentry, which reserves space for the local variable n (see below).

In case of gier drum or gier proc with tape input further reservations are made by reserve array. The length contained in the label on the tape is stored in the last of the cells reserved in this way.

Stack picture:

last used:... Input from tape stored
... in this array.

p-2: length : Length of array.

p-1: n: Number of parameters or, in case of gier drum, 0.

p : sr: Normal blockinformation.

p+1: i: Running address during input, initialized by reserve array.

return: Return address during call of take gier formal.

p+2: Normal return information.

p+3: First parameter.

... Following parameters.]

b k=c60+e70, i=10, a20, b10

d a=i

```
a1: ncn(c47) V 3 ; gier drum: appetite:= appetite+3; R:= 0;
; if appetite ≠ 0 then alarm(⌊<param⌋); go to common;
a2: srn(c47) VD 1 ; gier proc: gier: R:= -appetite-1;
ps 15 , hh c55 ; common:
ps (c46) , gr s-3 ; n:= R;
hsc5, qq1021.29+c1-1; blockentry (1)local level:(1)
hh (c84) , pm p3 ;
qq V LA ; if split(store[p+3],40,41,blind) ≠ 1 then
hv ra3 LB ; begin
hs c22 ; take formal (store[p+3]);
bs (p-1) ; if n > 0 then
ps ra4 , hv (c36) ; begin s:= ref; go to location[UA] end;
pm (c36) , gm p3 ; store[p+3]:= store[UA] end;

a3: hs c2+1c60.39,qq7 ; go to from tape; Note: no symbolic address became of r-mark
; comment the following 4 word must be kept together
qq (c36) ; as they are referred to via s-1, s-1 and s+1 resp;
gs p1 , hs c22 ; take gier formal: return:= s; take formal(M);
a4: ps (p1) , hs s1 ; ref: s:= ref; go to location[return+1];
[s1]grf c37 , it p3 ; out: UV:= RF; s:= p+3+n;
ps (p-1) , it c37 ; UA:= address(UV);
pa c36 , hh c6 ; go to exit proc;
; comment a6 is referenced as track relative 19 in 8a16;
a6: srn c34 , ac p-2 ; tape is in: length:= length - 1;
ps ra4 , ncn(p-1) ; s:= ref;
is (c46) , hv s1 ; if n ≠ 0 then go to location[last used + 1];
pm p4 , hs c22 ; take formal (store[p+4]);
arn p -2 ;
nkf 39 , grf(c36) ; store[UA]:= float(length);
arn p1 , ck -10 ;
ar p-1 D ; pack (R, 0,9,address(length)-1, 10,19,i, 20,39,0);
hs c11 , qq e26.29; to drum (R);
ps p5 , hh c6 ; s:= p+6; go to exit proc;

a17:arn c41 V NKB ; kb on: R:= if NKB then false else true;
arn c40 ;
hv c14 ; go to end R expr;
; comment 32 words used on this track.
```

[14.9.65]

[gier drum, gier proc and gier, page 2.]

```
d i=a+40 ; start next track;
a13:it p-1 , pa c46 ; sum error: last used:= p-1;
a14:pm 1023 , vy 17 ; ident error: M:= outputsum; select type;
      sy 64 , sy29[red]; writecr;
      sy 55[g] , sy 57[i] ; write red text ({<gier>});
      sy 53[e] , sy 41[r] ;
      sy62[black], lync47 ; read a char;
      gm 1023 , ud c89 ; outputsum:= M; select punch;
      ; comment a15 is referenced as track relation 7 in a3;
a15:grn c68 , lyn c47 ; from tape: sum:=0;
      nc 17 , hv ra15 ; if read a char  $\neq$  17 then go to from tape;
      lyn c47 , ca 4 ; if read a char = 4 then
      hs ra11 , hv ra18 ; begin one word; go to compute length end
      hs ra11 , ps ra15 ; one word; one word; go to from tape;
      ; procedure one word; comment to R, marks to RC;
a11:lyn rb1 , pa rb2 ; begin R:=0; b1:= char:= read a char; b2:=0;
a12:ac (c68) D ; rep: pack (R,0,9,char); sum:= sum+char;
b1: pi -1 , tl -7 ; indicator:= b1; tl (-7);
b2: bt -1 t -100 ; b2:= b2-100; if b2 < -512 then tl (10)
      tl 10 , hh s ; else begin char:= read a char; go to rep end
      ly c47 , hv ra12 ; end;

a18:tk 20 , ga rb3 ; compute length:
      tk 3 , gt rb3 ; length:= round (split(R,20,29,blind)+
      tk 12 , is 1 ; split(R,30,39,blind) $\times$ 40+1);
b3: ar s-1 D -1 ;
      ck 10 , gr p-2 ;
      hs ra11 , pm p3 ; one word; M:= store[p+3];
      gr p-3 X MRC ; store[p-3]:= R and RC marks; R:= M;
      sr p-3 NZ ; if R  $\neq$  0 then R:= R - store[p-3];
      hv ra14 NZ ; if R  $\neq$  0 then go to ident error;
      qq p-2 , hs c50 ; reserve array (length);
      qq -1 , hv s2 ; comment reserve array sets i:= last used;
      pm p-3 XV IRC ; R:= store[p-3]; RC:= marks(store[p-3]); skip;
a16:hs ra11 , ; rep: one word;
      gr (p1) MRC ; store[i]:= R and RC marks;
      pmn(p1) DX 1 ; i:= i+1;
      nc p-2 , hv ra16 ; if i  $\neq$  p-2 then go to rep;
      lyn c47 , tk -7 ; R:= 0; pack (r, 14,23,read a char,
      ly c47 , tk -7 ; 7,16,read a char, 0,9,read a char);
      ly c47 , tk 14 ; if sum  $\neq$  split (R, 14,23,blind) then
      nc (c68) , hv ra13 ; go to sum error;
      hs c2+c60.39, qq 19 ; go to tape is in; Note: no symbolic address because of r-m
      qq [check sum] ; All 40 words used; checksum for tracks stored here;

b k=e90, i=0 ; prepare loading of pass 6 words and
d i=e89, a1=a1-a, a2=a2-a, a17=a17-a; procedure names
      qq 143.9+c60.19+a1.29 ; pass 6 word for gier drum
      t gier drum; ; identifier
      qq 144.9+c60.19+a2.29 ; pass 6 word for gier proc
      t gier proc; ; identifier
      qq 141.9+c60.19+a2.29+112.39 ; pass 6 word for gier
      t gier; ; identifier
      qq 142.9+c60.19+a17.29+91.39f ; pass 6 word for kb on
      t kb on; ; identifier
d e71=4e71, e72=2e72, e89=i ; count number of tablewords
e ;
```

[14.9.65]

```
d c60=2c60                ; increase running track
e                        ; end a to a20 and b to b10
d e73=e73-e0, e75=e75-e0  ; Convert track numbers for ln exp and ↑integ
d e77=e77-e0            ;   to relative track numbers (relative to 1. st.
                        ;                          proc. track).
```

[Track reservations and GP come next]

[4. febr. 65]

[General pass administration, page 1]

[Track reservations for standard identifier tables.

These tracks are filled by the Translator Loading Administration]

```
d e98=c60                ; 1st track following std.procs.
d c60=c60+e85            ; reserve e85 tracks after std-procs.
d e81=c60                ; first track of std. ident. tables.

b k=c60+e70,  i=0        ; Reserve tracks for supplementary table.
d i=i+54e72,  c60=k-e70  ; e72+15 words.
d e88=c60-e81            ; number of supplementary tracks.
e                        ;

b k=c60+e70,  i=0        ; Reserve tracks for primary
d i=i+e71+39,  c60=k-e70 ; standard identifier table, containing
e                        ; e71 identifiers

b k=c60+e70,  i=0        ; Reserve tracks for pass 6 table.
d i=i+e71+39,  e83=c60   ; e29 = number of tracks for primary
d c60=k-e70             ; identifier table = number of tracks
d e29=c60-e83           ; for pass 6 table.
e                        ;
```

d e14=c60, c60=9e14, e15=c60, i=e13-39;
b a20 ;

[FORWARD PASS]

[Use of output: Place byte in R pos. 0-9, pos 10-39 cleared,
 40-41 any thing, call by hs e3. M is unchanged upon
 exit. If KB=1 then testprint

Call for input: arn (e1)+1 or pm (e1) X 1
 hs e2 LA hs e2 LA
 e1 holds the address of the last byte taken]

```
[OUTPUT]
e3: hv      a10      t  2      NKB ; reset by pack fourth
a10: hs      e6      ; testprint if LKB
     hv      (e3)     t  2      ; return from testprint
[OUTADDRESS]
e12: gr      82e13   t  1      ; pack first
     hr      s1      ;
     ck      -10     ,  gt (e12) ; pack second
     hr      s1      ;
     ck      20      ,  ac (e12) ; pack third
     hr      s1      ;
     ck      10      ,  ac (e12) ; pack fourth
     pa      e3      t  a10     ;
     hr      s1      NA      ; return if track not filled
[DRUMOUT]
     arn (7e4)      D  1      ;
     ca (4e4)      ,  nt (5e4) ;
     vk (7e4)      ,  is (e12) ;
     sk s-40      ,  arn 1e4   ;
     ca (5e4)      ,  hs  e5    ; alarm (‡<program too big‡)
     arn (e12)     ,  qq  e30   ;
     qq (e12)     t  -82      LB ;
a11: qq (1e4)     t  1      ; used:=used+1
e11: hv      e12     ;
[DRUMIN]
a12: arn (6e4)     D  1      ;
     ca (4e4)     ,  nt (5e4) ;
     vk (6e4)     ,  is (e2)  ;
     lk s-40     ,  arn (e2)  ;
     qq (e2)      t  -82      LB ;
     qq (1e4)     t  -1      ;
[INADDRESS]
e2: arn      81e13   t  1      ; unpack next word
     ga      e13-4   V      NA ;
     hv      a12     ;
     tk      10      ,  ga  e13-3 ;
     tk      10      ,  ga  e13-2 ;
     tk      10      ,  ga  e13-1 ;
[BYTEADDRESS]
e1: arn      e13-1   t  -4      ;
     hr      s1      ;
[-4] qq      ; BYTEBUFFER [1]
[-3] qq      ; [2]
[-2] qq      ; [3]
[-1] qq      ; [4]
[e13:BASEIN]
     qqf      ,  qq      ; [5]
```

[BACKWARD PASS]

```

[e3] hh    e3      t    2          NKB ;
     hs    e6              ;
[+2] hv    a11     ,    ck    10      ;
[e12] gr    123e13  t    -1          ;
[+4] hr    s1      ,    ck    20      ;
     ac    (e12)              ;
[+6] hr    s1      ,    ck    -10     ;
     gt    (e12)              ;
[+8] hr    s1      ,    ac    (e12)   ;
     pa    e3      t    e3          ;
     hr    s1              NA      ;
[8e12]arn  (7e4)    D    -1          ;
     ca    (10e4)   ,    it    (5e4)  ;
     vk    (7e4)    ,    is    (e12)  ;
     sk    s1      ,    arn    1e4    ;
     ca    (5e4)    ,    hs    e5      ;
     arn    (e12)   ,    qq    e30     ;
     qq    (e12)    t    82          LB ;
     qq    (1e4)    V    1            ;
[a11] hh    (e3)    t    2            ;
[e11] hv    e12              ;
[a12] arn    (6e4)   D    -1          ;
     ca    (10e4)   ,    it    (5e4)  ;
     vk    (6e4)    ,    is    (e2)    ;
     lk    s1      ,    arn    (e2)    ;
     qq    (e2)     t    82          LB ;
     qq    (1e4)    t    -1          ;
[e2] arn    42e13   t    -1          ;
     ga    e13-1    V              NA ;
     hv    a12              ;
     tk    10      ,    ga    e13-2   ;
     tk    10      ,    ga    e13-3   ;
     tk    10      ,    ga    e13-4   ;
[e1] arn    e13-1   t    -4          ;
     hr    s1              ;
[-4] qq              ;
[-3] qq              ;
[-2] qq              ;
[-1] qq              ;
[e13] qqf          ,    qq          ;

```

[GP. Page 4. To track 3e14 by TLA]

[This track is called in to 165e13 by initialize translator and it stays in core during all passes.

It holds the general routines and translator parameters

The translator parameters are first loaded in TLA to e 4 ff]

```
d i=165e13 ;
e6: bt 100 t -55 ; testprint: print count:= print count + 1;
; if print count > 10 then
e7: hs e8 ; new line print: newline;
e9: gm 41e13 , ga a2 ; print: save M:= M; save Raddr:= Raddr;
ck 0 , tl -30 ; M:= Raddr; digits:= -2;
pa a3 X -2 ; R39:= MX10-3; zero:= 0;
mln a4 , pt a5 ; next: char:= entier(R39); R39:= R39 - char;
a8: ck -10 , ga a6 ; if char = 0 then
a5: pa a6 V 0[or16]LZ ; begin char:= zero;
pt a5 V 16 ; if digits = 0 then zero := 16 end
ca (a3) , hv r-1 ; else zero:= 16;
a6: sy -1 , ncn s-e3-1 ; outchar(char);
e10: pa e6 t 100 ; if -, call from output then printcount:= 10;
a3: bt -1 t 1 ; digits:= digits + 1;
a2: pmn -1 DXV ; if digits ≤ 1 then
mln a7 , hv a8 ; begin R39:= R39×10; go to next end;
pm 41e13 , hv a1 ; Raddr:= save Raddr;
; M:= save M; go to space exit;
e8: sy 64 , pa e6 ; newline: outchar(<CR>); printcount:= 0;
a: sy 1 , pa r ; outchar(passno 1); pass nol:= 0;
a1: sy 59 , pa r ; space out: outchar(point); point:= 0;
sy 0 , hr s1 ; outchar(0); exit;
a4: 10-3 ; comment
a7: qq 10.39 ; constants for print;
[Meaning of message parameters:
Parameter 1, choice of medium Parameter 2, color, line, stop
0 Normally as 8, if n then as 12 0 red, line, stop immediately
4 ( ) w in input and output 1 s - - - after pass 6
8 n As 12 with w added 2 r black, no line, no stop
12 n() As typed by operator 3 p - , line , - -]
e5: vk 5e14 , it 123e13 ; message: tk:= messtrack;
bs (e12) , it -41 ; if place for mess ≤ outaddr then
a9: lk 124e13 , vk 6e14 ; place for mess:= place for mess - 41;
; to core(tk, place for mess); wait drum;
e94: hv (a9) , bs 1 ; tk:= first text track; go to core for message;
; comment ok is set false (0) by red message;
; endpass 6: e94h:
; if -, ok then go to init transl;
e92: vk 4e14 , hh e93 ; endpass: tk:= endpasstrack; go to comm;
e93: vk e14 , lk e25 ; inittransl: tk:= init transl track;
vk c61 , hv e25 ; comm: to core(tk, endpass place);
; wait drum; go to core endpass;
e4: [Translator parameters, ;
see TLA] ;
```



```

b k=5e14+e70, a6, b7, d40 ; start text block;

d b=i-1 ; used for clearing track at end;

pa a9 t 124e13 ; message track: restore place for message;
sy 58 , it 41e13 ; outchar(<LC>);
bs (e2) , it -41 ; if inaddress ≥ place for text then
b1: lk 42e13 , arn s1 ; place for text:= place for text - 41;
; to core(tk, place for text);
gt rb4 , tk 37 ; mess no:= part2(store[s+1]);
pt rb2 V 9e4 NO ; by select:= bits(36, 37, store[s+1]);
ud 8e4 LT ; by:= if byselect = 0 then error by
tk 2 V LT ; else if by select = 1 then typewrby
b2: tk 2 , ud 8e4 ; else if by select = 2 then alarm by
; else normal by;
hs e8 ; new line; comment cleared during use by init transl.
ps e93-1 LZ ; act:= bits(38, 39, store[s+1]);
pt e94 NT ; finis:= act= 0;
pa rb3 V 29 NT ; red := act < 2;
hv ral LO ; if act = 2 then go to print text;

b3: sy 62 , sy 35[l] ; outchar(if red then <red> else <black>);
sy 57[i] , sy 37[n] ; outtext(‡<line‡);
sy 53[e] , arn e4 ; print(CRcount);
hs e9 ;

a1: vk 7e14 , ud e10 ; printtext: tk:= text track 2; print count := 10;
b4: is (rb1) , it s-1 ; text addr:= place for text + mess no;
a2: ; go to test textaddr;
b5: pmn 0 X 1 ; next word: textaddr:= textaddr + 1;
is (rb5) , it s-40 ; test text addr: if textaddr - 40 ≥ place for text then
bs (rb1) , hv ra3 ; begin to core(tk, place for text);
lk (rb1) , vk 0 ; wait drum; textaddr:= textaddr - 40;
pmn(rb5) X -40 ; end;
a3: cl 36 , pa rb7 ; R5:= bits(36, 41, store[textaddr]);
ar 32 D LA ; M:= bits(0, 35, store[textaddr]);
ar 16 D LB ; count:= 7; next char:
a4: ca 160 , hv ra5 ; if R5 = <end text> then go to end;
b6: sy 58 , ck -4 ; outchar(char); Raddr:= R5;
ca 63 , arn e8 ; if Raddr = 63 then Raddr:= <CR>;
ga rb6 , it -80 ; char:= Raddr; count:= count - 1;
b7: bt -1 , hv ra2 ; if count = 0 then go to next word;
cln -6 , hv ra4 ; R5:= bits(34, 39, M); M:= bits(0,33, M);
; go to next char;
a5: sy 62 , vk (6e4) ; end: outchar(<black>);
lk (rb1) , vk (6e4) ; to core(inputtrack, place for text);
vy (8e4) ; by:= normal by;
a6: grn rb V 1 M ; clear this track for marks;
hrs1 ; if finis then go to init transl
hv ra6 ; else exit;

```

[GP. Page 6]

[Text tracks, 6e14 and 7e14]

[Comment: All texts must end in LC]

[Used in pass]

d:	tprogram too big;	; 1-8
d1:	tcharacter;	; 1
d2:	tcompound;	; 1
d3:	tpause;	; 1
d4:	t)<improper>;	; 1
d5:	tcomment;	; 1
d6:	tstring;	; 1
d7:	ttoo many identifiers;	; 2
d8:	tstack;	; 3
d9:	tend;	; 1
d10:	tsubscript;	; 6
d11:	tproc. call or ident.;	; 6
d12:	ttype;	; 6
d13:	ttoff;	; 1
d14:	ton;	; 1
d15:	t+declar.;	; 5
d16:	t+specif.;	; 5
d17:	t-declar.;	; 5
d18:	t-specific.;	; 5
d19:	tvalue;	; 5
d20:	t-formal;	; 5
d21:	t-delimiter;	; 3
d22:	toperand;	; 3
d23:	tdelimiter;	; 3
d24:	t-operand;	; 3
d25:	tnumber;	; 3
d26:	ttermination;	; 3
d27:	tsum;	; 1

```

d28:t
algol
;
d i=i-e69-e69
t
cdc algol
;
d i=i+e69+e69-2
d29:t
algol KA.
;
d i=i-e69-e69-e69
t
cdc algol KA.
;
d i=i+e69+e69+e69-3
d30:tfrom reader;
d31:twrong tape;
d32:t
algol KB.
;
d i=i-e69-e69-e69
t
cdc algol KB.
;
d i=i+e69+e69+e69-3
d33:t
algol KC.
;
d i=i-e69-e69-e69
t
cdc algol KC.
;
d i=i+e69+e69+e69-3

```

[Define final names of texts. Relative to d]

```

d e30= d-d, e31= d1-d, e32= d2-d, e33= d3-d, e34= d4-d, e35= d5-d;
d e36= d6-d, e37= d7-d, e38= d8-d, e39= d9-d, e40=d10-d, e41=d11-d;
d e42=d12-d, e43=d13-d, e44=d14-d, e45=d15-d, e46=d16-d, e47=d17-d;
d e48=d18-d, e49=d19-d, e50=d20-d, e51=d21-d, e52=d22-d, e53=d23-d;
d e54=d24-d, e55=d25-d, e56=d26-d, e57=d27-d, e58=d28-d, e59=d29-d;
d e60=d30-d, e61=d31-d, e62=d32-d, e63=d33-d
e ; end of text block;

```

[Each pass loads its own pass word. The pass word values read in here will goto init translator]

[Comp. to core, read a pass. Track 8e14]

[Call comp to core:

start address for loading in core - 1 in p;

hs coreplace for comp to core + e23;

qq identification on tape (may be ()), <exit here with error. sum Raddr=1
ident R=0>

normal exit:

Program uses whole indicator, by must be set before entry]

```

b k=8e14+e70, a9, b9      ; start block;

a:  pp 204e13 , it  2      ; test for pass 2:  p:= pass core place -1;
    bs (9e4)  , hv  ra2    ;  if passno > 2 then go to comp to core;
a1:                                     ;
b1:  qq -1     , ud  8e4    ; pass 2 to core:  after error:
    lyn rb1    , vy (8e4)  ;  by:= typewr by;  read a char;  by:= normal by;
a2:  pa rb6    , gs  rb9    ; comp to core:  i:= 0;  save s:= s;
a3:  pa rb2    , lyn rb1    ; from tape:  sum:= 0;
    nc 17      , hv  ra3    ;  if read a char ≠ <<> then go to from tape;
    lyn rb1    , ca  4      ;  if read a char = 4 then
    hs ra4     , hv  ra6    ;  begin one word; go to compute n end;
    hs ra4     , ps  ra3    ;  one word; one word; go to from tape;
a4:  lyn rb3    , pa  rb4    ; procedure one word; comment to R, marks to RC;
a5:                                     ;  begin r:= 0; b3:= char:= read a char; b4:=0;
b2:  ac -1     D          ;  rep: pack(R, 0,9,char); sum:= sum+char;
b3:  pi -1     , tl  -7     ;  RC:= bits(8,9,b3); tl(-7);
b4:  bt -1     t  -100     ;  b4:= b4-100; if b4 < -512 then bl(10)
    tl 10      , hh  s      ;  else begin char:= read a char; go to rep end
    ly rb1     , hv  ra5    ;  end one word;
a6:  tk 20     , ga  rb5    ; compute n:
    tk 3       , gt  rb5    ;  n:= bits(20,29,R)+bits(30,39,R)×40;
    tk 12      , ac  rb5    ;
b5:  qq -1     t  -1      ;
    hs ra4     , ps (rb9)   ; read and test identification: one word; s:= save s;
    nc (s1)    , hhn s1    ;  if Raddr = ident then
                                ;  begin R:=0; error exit end;
a7:  hs ra4     , ps (rb9)   ; read words: one word; s:= save s; i:= i+1;
b6:  gr p-1     t  1 MRC    ;  pack(store[p+i], o,39,R, 40,41,RC);
    pmn(rb5)    DX -1      ;  n:= n-1; M:= 0;
b7:  nc 1       , hv  ra7    ;  if n ≠ 1 then go to read words;
    lyn rb1     , tk  -7     ; read and check sum;
    ly rb1      , tk  -7     ;  if sum = read achar +
    ly rb1      , tk  14     ;  128×last 3 bits(read a char)+
    ca (rb2)    , hr  s2     ;  0×read a char then normal exit;
    arn rb7     , hh  s1     ;  Raddr:= 1; error exit;
    qq          ;  comment 2 unused words;
    qq          ;
a8:  pt rb8     V  e57 NZ   ; error:
    pt rb8     t  e61      ;  message (if R=0 then {<wrong tape>
    hs e5       ;  else {<sum>, 8+2});
b8:  hh ra1     , qqnr-1    ;  go to after error;
a9:                                     ; read a pass: comment this track is read in by
b9:  qq -1     , hs  e5     ;  endpass and entered here;
    hs ra      ,qqe60+10.29;  message({<on paper>, 8+2);
    qq (9e4)   , hv  ra8    ; comment ident , error exit: go to error
                                ;  normal exit: continue in end pass
d e23=a2-a      ;  define normal entry to comp to store
e                ; end comp to core block and track
e                ; end GP.
d e24=e15       ; Define final track of passes
s

```

[13.3.64]

[Pass 8. Page 0]

b k=e15+e70, i=205e13, a70, b40, d33; start pass 8 block
d e17=i ;

 ; comment variables
b: qq 2c1 ; initial last used
b1: qq ; instr
b3: ps (-1) ; ps instr
b8: qq -124e13.19-1.39 ; base
b12:qq ; const

b a11, b10, d4 ; Move pass 7 output to first track;

d d1=e25-80 ; bit array done[0:959]; comment 30 words×32bits
d b3=30d1 ; holds i.14
d b4=31d1 ; - j.14
d b2=3b4 ; - d.14
d d2=8b4 ; initial value for buffer 1
d d3=40d2 ; - - - - 2
[1b4 holds used tracks.14
 4b4 - last track.14
 5b4 - available tracks.14
 7b4 - output track.14]

a10:arn b4 ; boolean procedure move;
 ga b9 , tk 10 ; begin
 ck -5 , ga b8 ; move:= -,done[j];
b9: arn -1 t d1 ; done[j]:= true ; comment true is 1;
b8: ck -1 ; end;
 hr s1 LO ;
 ar 512 D ;
 ns (b8) , ck s ;
 gr (b9) , hr s2 ;

a11:arn b4 ; procedure track;
 ar 7b4 , sr 4b4 ; begin
 sr 5b4 NT ; tk:= j+outputtrack;
 ar 4b4 , ck 5 ; if tk > lasttrack then
 ga r1 ; tk:= tk-available;
 vk -1 , hh s ; end;

d4: qq 1.14 ; comment constant for counting;

a59: ; Start pass 8: move tracks:
a1: arn(1e4)D ; comment these instructions shift the
 ck -5 , gr 1b4 ; values of used tracks, lasttrack
 arn(5e4)D ; available and outputtrack so that they
 ck -5 , gr 5b4 ; are stored with unit in bit 14 instead
 arn(7e4)D ; of bit 9;
 ck -5 , gr 7b4 ;
 arn(4e4)D ;
 ck -5 , gr 4b4 ;
 sr 5b4 , sr 7b4 ; R:= lasttrack-available-outputtrack
 ar 5b4 LT ; if R < 0 then R:= R+available;
 hv a6 LZ ; if R ≠ 0 then
 gr b2 , ps 32 ; begin d:= R;
 ps s-1 , grn sd1 ; for s:= 0 step 1 until 959 do
 bs s , hv r-1 ; done[s]:= false; i:= j:= 0;

```

                                ;   loopi:
a2:  hs  a10                    ;   if move then
    hv  a5                      ;   begin
    hs  a11 , lk (b5) ;         track:  from drum(buffer1);
a3:  arn b4 , ar  b2 ;         loopj:
    gr  b4 , sr 5b4 ;         j:= j+d-(if j+d ≥ available then
    gr  b4   NT   ;             available else 0);
    arn b4 , sr 1b4 ;         if j < usedtracks then
    hs  a10   LT   ;         begin
    hv  a4                      ;         if move then
b5:  pp  d2 , it (b10);         begin  p:= buffer1;
    pa  b5 , gp  b10 ;         buffer1:= buffer2;  buffer2:= p;
    hs  a11 , lk (b5) ;         track; from buffer (buffer1);
b10:sk  d3 , hv  a3 ;         to drum (buffer2);  go to loopj
                                ;         end
                                ;         end j < used tracks;
a4:  hs  a11 , sk (b5) ;         track;  to drum (buffer1)
                                ;         end;
a5:  arn d4 , ar  b3 ;         i:= j:= i+1;
    gr  b3 , gr  b4 ;         if j < used tracks then
    sr  1b4 ;         go to loopi
a6:  hv  a2   LT   ;         end R ≠ 0;
e                                ; end move tracks

                                ; initialize pass 8:
vk(4e4) , lk124e13;  to core(last track, first output buffer);
it(10e4), pa  6e4 ; input track:=
it(1e4) , qq (6e4);   used tracks + first track - 1;
qq(5e4) t  1          ;   available:= available+1;
pm(4e4) DX 1          ;   last track:= output track := last track + 1;
ga 7e4 , ck  10       ;   track base:= last track-1;
ac  b8 , hs  a9       ;   call(next byte);
ac b37 , it(b37) ;     stack ref:= display+byte;
pt b17 , ac  b        ;   initial last used:= stack ref + 2;
pt e12 t -41          ;   adjust OUTADDRESS counting;
pa 2e4 t  d21          ;   min free stack:= start free stack;
grn d8   MA           ;   A[start stack 2]:= 1;
arn 2e4 , ck  10       ;   convert relative track number to
tl -30 , ps  d1        ;   absolute for table words
; pm 256 D  LB         ;   call ln, call exp, and call integ;
qq                                ;   MOGENS HACK
ac s13 , tl  1         ;
tk  -1 , ac s11        ;
ac s12 , arn7e4        ;
tk  12 , ac  r1        ;   t:= textno × 4 -5;
pp  -5 , ps  a21       ;   set return(next);
tk  -2 , ac  b10       ;   wconst:= Firstword - 1 + (39 + textno);
vk  0 , pi  0          ;   oldhalf:= sets:= false;  oldA:= oldB:= 0;
hv  a13                ;   go to byte read;

```

```

[addbyte and store: prel: parentprel: absaddr: with byte]
a1:  hs    a9                                ; R:= call(nextbyte);
[add R and store]
a2:  ac    b1    V                            IPC; instr:= instr + R; PA:= newA; PB:= newB;
[store: fixed instr:]                        ; skip;
a3:  pm    b1                                IPC; PA:= newA; PB:= newB;
     hv    a60                                NRA; if oldhalf then
     hv    a61                                NTB; begin if newhalf^
     qq                V                    NQA; -, (oldA = 1 ^
     hv    a61                                LPB; PB = )^
     qq                V                    NQB; -, (oldB = 1^
     hv    a61                                LPA; PA = 1) then
     pi    48    V - 51                    LQB; begin if oldA = 1 then PB := 1;
     pi    32    t - 35                        ; PA := 1; oldhalf:= false;
     arn    (b2)    ,    ck    -10        ; word[w]:= bits(0, 29, word[w]) +
     ar    b1                                IQC; instr; oldA:= newA; oldB:= newB;
b2:  gr    164e13                            MPC; A[w]:= PA; B[w]:= PB;
     pp    p2    ,    hr    s1            ; t:= t + 2; return
[not compatible:]                            ; end;
a61: pp    p2    V                            NRB; if -, sets then t:= t+2
     hs    a6                                ; else call(storeps)
                                           ; end;
a60: pp    p2    V                            NTB; if -, newhalf then t:= t + 2
     pi    2    V - 35                        ; else begin PA:= 0; oldhalf:= true end;
     pm    r                                IRA; if -, newhalf then oldhalf:= false;
     pm    b1                                IQC; w:= w - 1; oldA:= newA; oldB:= newB;
     gm    (b2)    t - 1                    MPC; word[w]:= instr; A[w]:= PA; B[w]:= PB;
     pp    p2    ,    hr    s1            ; t:= t + 2; return;

[maybe storeps:]
a5:  arn    b3    ,    ca    (b4)        ; if bits(0, 9, psinstr) = news then
     hr    s1                                ; return;
[storeps:]                                    ; if oldhalf then
a6:  pi    8    V - 12                    LRA; oldA:= 1 else oldA:= oldB:= 0;
     pin    2    V - 16                    ; sets:= false; oldhalf:= -, oldhalf;
     arn    (b2)    ,    ck    -10        ; R:= (if -,oldhalf then bits(0,29,word[w])
     ar    b3    ,    pp    p1            ; else 0) + psinstr; t:= t + 1;
     it    -1                                LRA; if oldhalf then w:= w - 1;
     gr    (b2)                                MQC; word[w]:= R; A[w]:= oldA; B[w]:= oldB;
     hr    s1                                ; return;

[left next byte:]
a7:  cl    10                                ; shift previous byte to M from left;
[next byte:]
a9:  arn    (e1)    t    1                ; Raddr:= input;
     hs    e2                                LA ;
     hr    s1                                ; return;

```



```

[maybechangetrack:]
a10: bs    (b5)    t    1          LTB; if -, newhalf  ∨ appetite + t > 1  ∨
      hv    a11                      ;
      pm    b1                      ; (newA = 0 ∧ newB = 0) then
      hv    a11          NC; go to change track;
      hr    s1          NRA; if -, oldhalf  ∨
      qq                      V      NQA; (oldA = 1 ∧
      hv    a11          LB; newB = 1)  ∨
      hr    s1          NA; (oldB = 1 ∧
      hr    s1          NQB; newA = 1) then return;

[changetrack]
a11: hs    a6          LRB; if sets then call(storeps);
      qq                      V      LOC; if changemode ≠ 3 then
      hs    a15                      ; call(takepoint 1);
      pi    32          t - 35        ; oldhalf:= false; PA:= 1;
      hs    8e12          ; DRUMOUT:
      qq    -1          , pm    b6      ;
      hs    a62          X              ; setnotlocal(stack 1, -1);
      qq    +1          , pm    b7      ;
      hs    a62          X              ; setnotlocal(stack 2, 1);
b18: srn    41          D              ; length:= -length
      ga    b18          , ac    b16    ; firstword:= firstword + length
      tk    -10          , ac    b33    ; w0const:= w0const + length
      ac    b19          , ac    b32    ; const0:= const0 + length; w0:= w0 + length;
      ac    b9           , ar    d10    ; lastword:= lastword + length;
b34: sc    b8           , pp    -161    ; relbase:= relbase + length; trackbase:=
b32: pa    b2           t      164e13   ; trackbase - 1; t:= -numberofwords×4 - 1;
b33: pa    b10          t      123e13   ; w:= w0; wconstant:= w0constant;
      hr    s1          LOC; if changemode = 3 then return;
      hs    a17          ; call(usestack 1);
      ar    d11          V      NOB; R:= R + (if changemode ≠ 1 then hsc2
      ar    d12          ; else hsc42);
      gr    (b2)          t - 1      MPC; w:= w - 1; word[w]:= R;
      pp    p4              ; A[w]:= PA; B[w]:= PB; t:= t4;
      hr    a21          LOA; if changemode = 2 then return to(next);
a63h: hr    s1           , it    (s-1) ; return;
b20: pm    -1           V      IPB; procedure setnotlocal(i, step);
a62: ga    b20          , hv    b20    ; value i, step;
      hr    s1          LA; a63h: if A[i] ≠ 1 then
      gm    (b20)        MPC; begin A[i]:= 1; i:= i + step; go to a63h
      hh    a63          ; end;
d10: qq    1.39          ;
d11: hs    c2            ;
d12: hs    c42

```

```

[lookupconstant:]
a14: pm    b12          IPC; PC:= Cconst;
      it    (b10)      ,    pt    b21  ; until:= wconst;
b19: pa    b11      t    123e13      ; constaddr:= const0
b22: gi    -1        D          ; PCconst:= PC;
      pa    b35      t    d9          ; lasttest:= false; more: constaddr:=
a64:b11: pm    -1      X    1          IPC; constaddr + 1; PC:= C[constaddr];
b21: bs    (b11)      t    -1          ; if constaddr > until then begin if
b35: hh    r -1      ,    hr    s1      ; lasttest then return elsegoto nextlist end;
      sr    b12          ; R:= word[constaddr] - const;
      hv    a64          NZ ; if R ≠ 0 then go to more;
      gi    r1      ,    arn    b22    ; if PC ≠ Cconst then go to more;
      nc    -1      ,    hv    a64    ; returnto(constin);
a65: hr    a30      ,    pa    b35    ; nextlist: lasttest:= true;
b9:  pt    b21      t    163e13      ; until:= lastword;
      it    (b2)      ,    pa    b11    ; constaddr:= w;
      hv    a64          ; go to more;
d d9= a65 - b35          ; define value for lasttest false

[takepoint 1:]
a15: pm    (b6)      DXV  1          ; Raddr:= stack 1:= stack 1 + 1;
[takepoint 2:]          ; skip;
a16: pm    (b7)      DX  - 1          ; Raddr:= stack 2:= stack 2 - 1;
      ga    b23          ; stackaddr:= Raddr;
      arn    (b2)      D          ; R:= 0; pack(R, 10, 19, w);
      ck    -10      ,    ar    b8      ; R:= R + base;
      qq          V      LRA; stack[stackaddr]:= R;
b23: gr    -1        V      M      ; A[stackaddr]:= 0;
      gr    (b23)      MB ; B[stackaddr]:= if oldhalf then 1 else 0;
b13: it    d21      t  - 1          ; freestack:= freestack - 1;
      bs    (2e4)          ; if freestack < minfreestack then
      bt    (2e4)      t  - 1          ; begin minfreestack:= minfreestack - 1;
      hr    s1      ,    qq    e38    ; if minfreestack < 0 then alarm(e38)
      ps    r-2      ,    hv    e5      ; end; return;

[takestack 1:]
a17: pm    (b6)      XV          IPC; R:= stack[stack 1]; PC:= C[stack 1];
[take stack 2:]          ; skip;
a18: pm    (b7)      XV          IPC; R:= stack[stack 2]; PC:= C[stack 2];
[unstack 1:]            ; skip;
a19: qq    (b6)      V  - 1          ; stack 1 := stack 1 - 1;
[unstack 2:]            ; skip;
a20: qq    (b7)      t    1          ; stack 2 := stack 2 + 1;
      qq    (b13)     t    1          ; freestack:= freestack + 1;
      hr    s1          ; return;

```

```

[initializereturn:]
a21: qq      , ps      a21 ; comment this instruction will ensure
[next:]                ; that the first return which does not match
a22: pmn (e1) X 1      ; a call always will goto next;
      hs      e2      LA;
      ga      b24      ; byte:= input;
[byteread:]
a13:b24: pmn 133 X d2    ; R:= table[byte]; actions:= bits( 0, 19, R);
      ga      r1      , gt b14 ; changemode:= bits(0, 1, R); destroy:=
      pi      -1      t 15    ; bit(2, R) = 1; newhalf:= bit(3, R) = 1;
      gr      b1      MPC; instr:= bits(20, 39, R);
      pa      b1      , pt b1  ; newA:= bit(4, R); newB:= bit(5, R);
      mb      15      D      ; appetite:= bits(6, 9, R);
      ga      b5      ;
      qq      V      NRB; if sets ^ destroys then call(storeps);
      hs      a6      LTA;
b5:   bs      p-1      ; if t + appetite > 0 then
      hs      a10      ; call(maybechangetrack);
      pm      (b24)    IPC; PC:= Ctable[byte];
      hh      b14      NPB; if PB ≠ 0 then
      arn     b1      , ck -10 ; begin if PA = 0 then instr:=
      ga      b26      ; auxiliarytable[bits[30,39,instr]] else
      pmn     (b26) XV  NPA; begin pack(instr,0,9,bits(30,39,instr));
      tk      10      , ar b26 ; pack(instr, 30, 39, 0) end;
b14h: gr      b1      , hv -1  ; end;
      b26: qq      -1  ; gotolabel(action);

[withoperand:]
a23: pmn (e1) X 1      ;
      hs      e2      LA ; R:= operandtable[input];
      ga      r1      ; operandaction:= (bits 10, 19, R);
      pmn     -1      X d2  ; pack(R, 10, 19, 0);
      gt      b15      , mb d14 ; instr:= instr + R;
b15h: ac      b1      , hv -1  ; gotolabel(operandaction);
d14: qq      -1.9 + 1.19 - 1.39 ;

[constantoperand:]
a24: hsn      a9      X      ; M:= 0; call(nextbyte);
      hs      a7      ; call(leftshiftbyte);
      hs      a7      ; call(leftshiftbyte);
      hs      a7      ; call(leftshiftbyte);
      cl      -30      ; shift read in constant right to R;
      gr      b12      MB ; const:= R; Aconst:= 0; Bconst:= 1;
      arn     b12      , sr d15 ; if const = 1.0 then
      pa      b1      V c40 LZ ; pack(instr, 0, 9, addrof 1)
      sr      d16      ; else if const ≠ -1 then
      hv      a29      NZ ; go to useconst;
      srn     d17      , hv a2  ; R:= -bit 28; go to addRandstore;
d15: qq      t 256      ;
d16: qq      -1      t 256 ;
d17: qq      r          ;

```

```

[sreloperand:]
b4: a25: paf  _1  D   c1           ; news:= display; comment f is used later;
      hs    a9                ; call(nextbyte);
      sc    b4                ; news:= news - Raddr;
      hs    a5                LRB; if sets then call(maybestoreps);
      pm    1    DXV          LTB; appetite:= appetite +
      pm    3    DX           ;    (if newhalf then 1 else 3);
      ac    b5    ,    bs    (b5) ; if t + appetite > 0 then
      hs    a10                ;    call(maybechangetrack);
      pp    p-1                LRB; if sets then t:= t - 1;
      pm    r                  IRB; sets:= false;
      hs    a1                ; call(addbyteandstore);
      pm    b4    X            IRB; pack(psinstr, 0, 9, news);
      ga    b3    ,    pp    p1 ; sets:= true;
      hr    s1                ; return;

[condgoto 1:]
b6: a26: pmn  d7    XV          IPC; R:= stack[stack 1]; PC:= C[stack 1];
[condgoto 2:]
b7: a27: pmn  d8    X          IPC; R:= stack[stack 2]; PC:= C[stack 2];
[condgoto:]
a28: hv    a33                NPA; if PA = 0 then go to localgoto;
      ar    d11                ; const:= R + hsc2;
      gr    b12                MPC; Cconst:= PC;
[useconstant:]
a29: hs    a14                ; call(lookupconstant);
      pm    4    DX           ; const not in: appetite:= appetite+4
      ac    b5    ,    bs    (b5) ; if t + appetite > 0
      hs    a10                ; then call(maybechangetrack);
      pp    p4                ; t:= t+4;
      pm    b12                IPC; wconst:= wconst + 1;
b10: gm    162e13 t    1      MPC; word[wconst]:= const; C[w]:= Cconst;
      it    (b10) ,    pa    b11 ; constaddr:= wconst;
[constin:]
a30: hs    a3                ; call(store);
      arn   b11    ,    sr    b2 ; pack(word[w], 0, 9, constaddr - w);
      ga    (b2) ,    hr    s1 ; return;

[uncondgoto 1:]
a31: pmn  (b6) X              IPC; R:= stack[stack 1]; PC:= C[stack 1];
[undcondgoto:]
a32: hv    a34                LPA; if PA = 1 then go to nonlocalgoto;
[localgoto:]
a33: ck    10                ; constaddr:= bits(10, 19, R) +
b16: ar    124e13 D          ; firstword;
      ga    b11    ,    arn   d18 ; if PB = 1 then
      ac    b1                LPB; instr:= instr + (hh - hv);
      hv    a30                ; go to constin;
d18: qq    4.25                ;

```

```

[nonlocalgoto:]
a34: ar    d11                ; const:= R + hsc2;
    gr    b12                MPC; Cconst:= PC;
    hs    a14                ; call(lookupconstant);
    pm    b12                IPC; notthere: instr:= const;
    gm    b1                 MPC; Anew:= Aconst; Bnew:= Bconst;
    pm    8      DX          ITB; newhalf:= false;
    ga    b5      ,   bs    (b5) ; t:= t + 4; if2appetite + t > 0 then
    hs    a10                ; call(maybechangetrack);
    hv    a3                 ; go to store;

[end block:]
a35: hs    a36                ; call(endproc);
    pp    p-2      ,   pm    r    ; t:= t - 2; comment for right halfword;
    hr    s1                 IRA; oldhalf:= true; return;

[end proc:]
a36: hs    a9                 ; call(nextbyte); stack 1:= stack 1 + 1;
    gr    (b6)    t    1      M ; stack[stack 1]:= R; C[stack 1]:= 0;
    hs    b13                ; call(checkfreestack);
    hv    a3                 ; go to store;

[releasefor:]
a37: hs    a19                ; call(unstack 1);
[if:]
a38: hs    a20                ; call(unstack 2);
    hv    a20                ; go to unstack 2;

[endelse:]
a39: hs    a16                ; call(takepoint 2);
    hv    a16                ; go to takepoint 2;

[bypass abs:]
a40: is    (b7)      ,   arn    s1    ; R:= stack[stack 2 + 1];
    hv    a32                IPC; PC:= C[stack 2 + 1]; go to undcondgoto;

[bypass LT:]
a41: is    (b7)      ,   arn    s1    ; R:= stack[stack 2 + 1];
    hv    a28                IPC; PC:= C[stack 2 + 1]; go to condgoto;

[goto bypass:]
a42: hs    a18                ; call(take stack 2);
    hv    a32                ; go to uncondgoto;

```

```

[programpoint:]
a43: hs    a17                ; call(takestack 1);
      qq      V                ; skip ;
[return information:]
a44: hs    a18                ; call(takestack 2);
[end local notsame track: ]
a12: pm    d27                IPA; PA:= 1;
      ac    b1                MPC; instr:= instr + R;
d27: hv    a3      ,          ; go to store;

[callstandardprocs: endcallstandard: standardproc:]
a45: hs    a9                ; call(nextbyte);
      ck    10      ,   ac    b1  ; pack(instr, 30, 39, Raddr);
[localswitch:]
a46: hs    a9                ; call(nextbyte);
      ck    -10     ,   hv    a2  ; Rcount:= Raddr; goto addRandstore;

[begincall:]
a47: hs    a9                ; call(nextbyte);
      ck    10      ,   hv    a2  ; pack(R, 10, 19, Raddr); pack(R, 0, 9, 0);
                                   ; goto addRandstore;

[enddo:]
a48: hs    a1                ; call(addbyteandstore);
      hv    a16              ; go to takepoint2;

[endsingledo:]
a49: hs    a3                ; call(store);
      hs    a16              ; call(takepoint2);
      it    (7e4)   ,   pa    (b7) ; pack(stack[stack 2], 0, 9,
      hv    a9                ; -previousouttrack);

[simpleblock 0: describedinstackblock0:]
b17:a50: pa    b25   V   c1          ; work 1:= stackref; skip;
[simple: describedinstack: endcall:]
a51: pa    b25                ; work 1:= 0;
      hs    a9                ; call(nextbyte);
      pa    b1      t    c1    ; pack(instr, 0, 9, display - Raddr);
      sc    b1                ;
      hs    a9                ; call(nextbyte);
b25: ar    -1      D          ; pack(R, 10, 19, work 1 + Raddr);
      ck    -10     ,   hv    a2  ; pack(R, 0, 9, 0); go to addRandstore;

[end local: ]
a70: hs    a18                ; call(takestack 1);
      hv    a12              LPA ; if PA = 1 then goto endlocal notsametrack;
      ck    10      ,   ud    b16  ; c0nstaddr:= bits(10,19,R) + firstword;
      ga    b11     ,   pm    d30  ; if PB=1 then begin if w=constaddr then
      pa    b24   V   d28      LPB ; begin byte:= d28; comment endlocal as
      pm    d31   V          ; halfword; return to(bytread) end
      ca    (b2)    ,   hr    a13  ; else instr:= endlocal hh end
      gm    b1                ; else instr:= end local hv
      hs    a3                ; call(store);
      nt    (b2)    ,   it    (b11) ; pack(word[w], 10, 19, constaddr - w);
      pt    (b2)    ,   hr    s1   ; return;

```

```

[beginlocal:]
a52: pmn   (b6)   X           ; R:= stack[stack 1];
      hs     a19           ; call(unstack 1);
      ck     10      ,   ac   b1   ; pack(instr, 30, 39, Raddr);
      arn    b4      ,   gt   b1   ;
      hs     a9           ; call(nextbyte);
      ck     -10     ,   sc   b1   ; pack(instr, 10, 19, display - Raddr);
      hv     a3           ; go to store;

[indexmult:]
a53: hs     a9           ; call(nextbyte);
      ga     b28         ; pack(mkfs, 0, 9, Raddr);
      hs     a22         ; call(next);
      acn    b1           M      ; newA:= newB:= 0;
      arn    b28     ,   sr   b1   ; word[w]:= word[w] - instr + mkfs;
      ac     (b2)     ,   hv   a3   ; go to store;
b28: mkf     s-1         ;

[tk 1:]
a54: bs     p516     ,   hv   a3   ; if t + 5 ≤ 0 then go to store;
      pp     1       ,   hr   s1   ; t:= 1; return;

[paramconstant:]
a55: pan    b27     X   3           ; j:= 3; M:= 0;
a66: hs     a9           ; notfinished: call(nextbyte);
      ck     10           ; R:= Raddr;
b27: bt     -1       t - 1         ; j:= j - 1; if j > -1 then
      cl     -10     ,   hv   a66   ; begin-bhifcrighccoMN+goco-nocfinished endN
      cl     30      ,   hv   a2    ; shift read in constant left to R;
                                   ; go to addRandstore;

[else:]
a56: hs     a20         ; call(unstack 2);
      hs     a16         ; call(takepoint 2);
      pa     b24     t   d3       ; byte:= 114; commend bypass abs;
      hr     a13         ; return to(byteread);

```

```

[takeforlabel:]
a57: arn    d11    ,    pm    d20    ; M:= hsc2 + stack[stack 2];
      ar    (b7)   X          IPC; PC:= Cconst:= C[stack 2];
      ar    d18          LPB; const:= if PB = 0 then hvs else hhs;
      gr    b12    ,    is    (b7)   ; enddotrack:= bits(0, 9, stack[stack2-1])-1;
      arn   s1      ,    ga    b29    ;
      mt    d16    ,    ar    (b7)   ; pack(const 0, 9, relpart(stack[stack 2]) -
      tk    10     ,    ga    b12    ; relpart(stack[stack 1]); R:= bit 29;
      tk    20     ,    ar    d19    ; if trackpart(stack[stack 2]) = trackpart
      nc    0       ,    gmn   b12    ; (stack[stack 2 - 1]) then begin const:= M;
      hs    a20          IPA; R:= 0 end; call(unstack 2);
      nc    (b29)   ,    hv    a67    ; if enddotrack  $\neq$  0 then go to single do;
      pa    b24     V    d5          LZ ; byte := if R  $\neq$  0 then gmp byte
      pa    b24     V    d4          ; else if PB = 1 then gmp byte MC
      qq    (b24)   t    1          LPB; else gmp byte MA;
      acn   b12          MPC; Cconst:= PC;
      hs    a13          ; call(bytread);
      pa    b24     t    d6          ; byte:= pm rrel const;
      hr    a13          ; returnto(bytread);
d19: qq    s        ,              ;
d20: hv    s        ,              ;

[singledo:]
a67: ac    b12          MPC; Cconst:= PC; const:= const + R;
      pm    b12    ,    arn   (b7)   ; M:= const; R:= stack[stack 2];
      it    (b16)   ,    pa    b30    ; oncurrenttrack:= A[stack 2] = 0; enddobase:=
      hh    a68          NA ; firstword; if -, oncurrent track then
      arn   (b29)   D    -1          ; begin if enddotrack - 1 = firsttrack then
      ca    (10e4)   ,    it    (5e4) ; enddotrack:= enddotrack - 1 +
b29: vk    -1        ,    arn   (b7)   ; number of tracks; selecttrack(enddotrack);
      nt    (b18)   ,    lk    (b30) ; enddobase:= enddobase - length;
      it    (b30)   ,    pa    b31    ; drumto(enddobase); selecttrack(enddotrack);
a68h: vk    (b29)   ,    gt    b30    ; end;
b30: gm    -1        t    - 1        MPC; word[enddobase + relpart(stack[stack 2])]:=
b31: sk    -1        LA ; const; Cconst := PC;
      hv    a9          ; if -, oncurrenttrack then todrum(enddobase);
                        ; go to nextbyte;

```



```

; end pass 8:
a58:hs a11 ; call(change track);
      arn 1e4 , ud 10e4 ; if pass inf then
      hs e7 ; begin
      arn (b2) , ud 10e4 ; new line print (used tracks);
      ck 10 , hs e9 ; print(relative start address);
      arn (b2) , ud 10e4 ; print (start track);
      ck -10 , hs e9 ; end;
      pm (b2) , ud 6e4 ; p:= last track of translated program;
      vk p , lk e25 ; word on track[p,0]:= startword;
      vk p , gm e25 ; word on track[p,1]:= initial drumplace:=
      arn 2e4 , gt b ; float(px40-1);
      cln -10 , nkf 9 ; pack (word on track[p,2],
      mkf d32 , srf d33 ; 0,9,initial last used,
      grf 1e25 , pm b ; 10,19,first track of std procs,
      gm 2e25 , sk e25 ; 20,39,0);
      , ud 8e4 ; if LKA then typechar;
      lyn _1 D LKA ;
      vy (8e4) ;
      hv a69 NKB ; if -, test output then goto end translation;
      pa 1e8 t 9 ; pass no 1:= 9;
      pa 2e8 t 59 ; point:= <.>;
      pp 204e13, hs e5 ; p:= pass place - 1;
      hv r1 ,qqe60+10.29; message({<from reader}, 8+2);
      ps (e93) , vk s8 ; to core(comp to core track, track place);
      lk e25 , ud 8e4 ; by:= type wr by;
      vk s8 , lyn b ; read a char;
      hv a69 NKB ; if -,test output then goto end translation;
      vy (8e4) , hse23+e25 ; comp to core(p) ident:(9) error:
      qq 9 , hv 33e25 ; (in comp to core);
      hs 205e13 ; call (output translated program);

a69: , ud 1e4 ; end translation:
      vk p6 , lk 0 ; initialize core 0 to 5;
      vk p5 , ud 6e4 ; to core(RS 5, last RS place);
      gp 0 , lk c67 ; set addr(L0) to:(last track of translated progr);
      vk , hv c23 ; go to start run;

f
d32: 40 ; comment constants for
d33: 1 ; computation of initial drumplace;
m

```

[Pass 8 uses 3 tables:

1. Auxiliary table (0d1 - 16d1). This table is referenced from the input table and contains those instructions which require more bits than are available in the input table, see below.
2. Operand table (1d2 - 7d2). This table is referenced by action a23 (with operand), which will read in another byte. This byte gives access to a word in the operand table. This word, cleared in bits 10-19, will be added to the instruction and pass 8 will proceed to the action address given in bits 10-19 of the word.
3. Input table (10d2 - 133d2). The format and other treatment of the instruction corresponding to each input byte value is given in this table as follows:

Track link and s-preservation

0.2 Normal link

1.2 - - , destroys s

2.2 Parameter link

3.2 - - , destroys s

4.2 Normal link, but instruction is suppressed

5.2 - - - - - , destroys s

6.2 Link is suppressed (unconditional jump)

7.2 - - - - - , destroys s

Instruction format and marks

0.5 Full word, no marks

1.5 - - , f (=b) mark

2.5 - - , comma (=a) mark

3.5 - - , comman and f (=c) mark

4.5 Half - , f indifferent

5.5 - - , f mark

6.5 - - , no-f mark

Appetite

Bits 6-9 supply the appetite which controls the test for transition to a new track.

Formation of non-zero bits of instruction

Marks on table-word	Bits of final instruction	are taken from
No-f	0-19 are cleared 20-39	20-39 of table-word
f and comma	0-9 10-19 are cleared 20-29 30-39 are cleared	30-39 of table-word 20-29 of table word
f, no-comma	0-39	address given in bits 30-39 which refer to auxiliary table

Action

Bits 10-19 of the tableword give the address of action-program to be performed]

[Auxiliary table				Input byte	Produced by]
d1:	mt	c33	LT ; 0d1	10	‡
	arn	512 D	NZ ; 1d1	20	=
	arn	(c36) , hs	c18 ; 2d1	25	goto computed
	pm	, hs	c22 ; 3d1	55	formal
	srf	c35 V	NT ; 4d1	56	:
	tkf	-29 , nkf	39 ; 5d1	58	:
	pm	, hs	c26 ; 6d1	59	index call
	pm	, hs	c38 ; 7d1	60	switch call
	qq	-1 , hv	s2 ; 8d1	124	array declaration
	arn	c46 , ck	-10 ; 9d1	125	- - , take last used
	qq	, hs	c50 ; 10d1	128	- - , reserve space
	hs	c10 , qq	e74+e73.29; 11d1	103	call ln
	hs	c10 , qq	e76+e75.29; 12d1	122	call exp
	hsn	c9 , qq	e78+e77.29; 13d1	130	call ‡integ
d30:	ps	(c84) , hh	s ; 14d1	internal	end local
d31:	ps	(c84) , hv	s ; 15d1	internal	end local
	ps	11 , hh	c55 ; 16d1	internal	initial byte

[Operand table				Byte value	Addressing]
d	d2=i-1				
qq	rc41+i, qq	a24	; 1d2	1	constant operand
b37:	qq <u>c1</u> , qq	a1	[2d2	2	abs addr, addresspart is changed to stackreference by start pass 8]
qq	p , qq	a1	; 3d2	3	p rel
qq	(p) , qq	a1	; 4d2	4	(p rel)
qq	s , qq	a25	; 5d2	5	s rel
qq	(c36) , qq	a3	; 6d2	6	(UA)
qq	c37 , qq	a3	; 7d2	7	UV
zq	t	8d2	; 8d2		not used
zq	t	9d2	; 9d2		not used

[Input				Input produced by]
[Input table				Input produced by]
qqf	5 t	a3 + d1.29;	10 mt c33 LT	‡
mtf	4.5+3 , qq	a3 +c33.29;	11 mt c33	(f) < ≤ ≥ >
qq	2.2+1.5+5 t	a3 ;	12 qqf t	blind in type proc
hvf	6.2+4.5+7 , qq	a3 +c12.29;	13 hv c12	(f) end UV expr
tkf	5.5+4 , qq	a3 +995.29;	14 tkf -29	round
hvf	6.2+4.5+7 , qq	a3 +c13.29;	15 hv c13	(f) end RF expr
hvf	6.2+4.5+7 , qq	a3 +c14.29;	16 hv c14	(f) end R expr
hvf	6.2+4.5+7 , qq	a3 +c15.29;	17 hv c15	(f) end addr expr
ab	5 DX	a3 ;	18 ab 0 DX	-, (not)
srf	6.5+4 , qq	a3 +c34.29;	19 sr c34	< > ‡
qqf	5 t	a3 +1d1.29;	20 arn 512 D NZ	in equal
arnf	(5.5+4), qq	a3 +c36.29;	21 arnf(c36)	value rounded
srf	5.5+4 , qq	a3 +c71.29;	22 srf c71	entier
nkf	5.5+4 , qq	a3 + 39.29;	23 nkf 39	float
hhf	7.2+4.5+7 , qq	a3 +c16.29;	24 hh c16	(f) goto non local
qqf	1.2+2.5+5 t	a3 +2d1.29;	25 arn(c36), hs c18	goto computed
hvf	7.2+4.5+7 , qq	a3 + c3.29;	26 hv c3	(f) goto local
pmf	4.5+3 , qq	a3 +c36.29;	27 pm c36	(f) clear UA
gsf	1.2+4.5+7 , qq	a3 +c36.29;	28 gs c36	(f) end addr expr
ann	5.5+4 t	a23 ;	29 annf <op>	abs
srn	5.5+4 t	a23 ;	30 srnf <op>	- < ≤ ≥ >
ar	5.5+4 t	a23 ;	31 arf <op>	until + array
mk	5.5+4 t	a23 ;	32 mkf <op>	x until ‡r array
ab	4.5+3 t	a23 ;	33 ab <op>	(f) ^
mb	4.5+3 t	a23 ;	34 mb <op>	(f) v

[Input table, continued			[Input byte generates		Input produced by]	
mb	5 X	a23	; 35	mb <op> X	≡	
sr	5.5+4 t	a23	; 36	srf <op>	- < ≤ ≥ > = ‡ until	
dk	5.5+4 t	a23	; 37	dkf <op>	/ ∴	
snn	4.5+3 t	a23	; 38	snn <op>	(f) =	
ann	4.5+3 t	a23	; 39	ann <op>	(f) ‡	
arn	5.5+4 t	a23	; 40	arnf <op>	array arithm expr	
arn	6.5+4 t	a23	; 41	arn <op>	boolean expr	
gr	5.5+4 t	a23	; 42	grf <op>	value rounded r:=	
gr	6.5+4 t	a23	; 43	gr <op>	b:= array	
grn	4.5+3 t	a23	; 44	grn <op>	(f) :=0	
pm	4.5+3 t	a23	; 45	pm <op>	(f) array := clear	
gm	4.5+3 t	a23	; 46	gm <op>	(f) array := clear	
ps	1.2+4.5+3 t	a23	; 47	ps <op>	(f) goto local goto nonlocal	
gr	1.5+5 V	a23 LA	; 48	grf <op> VLA	until, coupled to 49	
acn	9 t	a23 MA	; 49	acn <op> MA	until, coupled to 48	
gr	1.5+5 t	a23 M	; 50	grf <op> M	<expr>step	
grn	5 t	a23 M	; 51	grn <op> M	0step	
gm	5 t	a23 M	; 52	gm <op> M	<variable>step	
is	(5.5+6)t	a23	; 53	isf (<op>)	in 3rd, 4th..subscript	
mk	(5.5+4)t	a23	; 54	mkf (<op>)	in 2nd subscript	
qqf	1.2+2.5+5 t	a23+3d1.29;	55	pm <op> , hs c22	formal	
qqf	1.5+5 t	a3 +4d1.29;	56	srf c35 V NT	in int.divide	
arf	1.5+9 , qq	a3 +c35.29;	57	arf c35 t	in int.divide, two instr	
qqf	3.5+5 t	a3 +5d1.29;	58	tkf -29 , nkf 39	in int.divide	
qqf	1.2+2.5+5 t	a23+6d1.29;	59	pm <op> , hs c26	indexcall	
qqf	1.2+2.5+5 t	a23+7d1.29;	60	pm <op> , hs c38	switchcall	
grf	5.5+4 , qq	a3 +c54.29;	61	grf c54	in ‡int	
grf	5.5+4 , qq	a3 +c68.29;	62	grf c68	in ‡int	
gmf	4.5+3 , qq	a3 +c54.29;	63	gm c54	(f) in ‡int	
gmf	4.5+3 , qq	a3 +c68.29;	64	gm c68	(f) in ‡int	
pmf	(4.5+3), qq	a3 +c36.29;	65	pm (c36)	(f) in take value unrounded	
d26:						
pmr	4.5+3 t	a29	; 66	pm r<reladdr>	(f) internal in take forlabel	
sy	4.5+3 t	a1	; 67	sy <by>	(f) trykende, -slut etc	
grp	6.5+4 t	a1	; 68	gr p<by>	clear R	
grp	5.5+4 t	a1	; 69	grf p<by>	clear RF	
d24:						
gmp	4.5+3 t	a1	; 70	gm p<by>	(f) clear. internal take forl.	
qq	4.5 t	a53	; 71	mkf s<by>	3rd, 4th... subscript	
hsp	6.2+4.5+7 t	a48	[72	hs p<by>	(f) enddo	
				takepoint2]		
zq	6.2+ 7 t	a49	[73	zq	end single do	
				takepoint2 and -actual track]		
lynf	4.5+3 , qq	a3 +c68.29;	74	lyn c68	(f) lyn	
tkf	6.5+0 , qq	a54 + 1.29[75	tk 1	then while	
				never as last word on a track]		
hhf	2.2+3.5+5 , qq	a46+c20.29;	76	hhf c20 , qq <by>	local switch <by>	
qq	2.2+1.5+5 t	a55	[77	qq <by4>.9+<by3>.19+<by2>.29+<by1>.39		
				paramconst <by1><by2><by3><by4>]		
hsf	2.2+3.5+5 , qq	a47+ c5.29;	78	hsf c5 , qq <by>.29	beg call<by>	
hvf	6.2+2.5+9 , qq	a45+c21.29[79	hv c21 , qq <rel>+<track>.29		
				endcall standard <track><rel>]		
qq	2.2+2.5+5 t	a45	[80	qq , qq <rel>+<track>.29		
				param standard <track><rel>]		

	[Input table, continued	[Input byte generates	Input produced by]
ps (2.2+3.5+5), ps sa51		[81 psf(displ-<block>) , psf s<rel>	param simple <block><rel>]
qq 2.2+3.5+5 , ps a50		[82 qqf displ-<block> , psfstackref+<rel>	param simple block0 <block><rel>]
ps (2.2+2.5+5), pm sa51		[83 ps (displ-<block>) , pm s<rel>	param described in stack <block><rel>]
qq 2.2+2.5+5 , pm a50		[84 qq displ-<block> , pm stackref+<rel>	param described in stack block0 <block><rel>]
ps (6.2+2.5+9), psnsa51		[85 ps (displ-<block>) , psns<rel>	end call <block><rel>]
nkf 5.5+4 , qq a3 + 9.29;	86 nkf 9		floataddr (lyn)
hsnf1.2+2.5+5 , qq a45+ c9.29	[87 hsn c9 , qq <rel>+<track>.29		call standard no param <track><rel>]
hsf 1.2+2.5+5 , qq a45+c10.29	[88 hs c10 , qq <rel>+<track>.29		call standard 1 param <track><rel>]
hsf 1.2+2.5+5 , qq a45+c11.29	[89 hs c11 , qq <rel>+<track>.29		call standard 1 Rparam <track><rel>]
hhf 6.2+4.5+7 , qq a36+ c6.29	[90 hh c6		(f) end proc <appetite>
hhf 6.2+4.5+7 , qq a36+ c7.29	[91 hh c7		(f) end typeproc <appetite>
hsf 1.2 +5 , qq a35+ c8.29	[92		, hs c8 (f) end block <appetite>
hsf 2.2+2.5+5 , qq a52+ c5.29	[93 hs c5 , qq displ-<block>+appet.fromstack1.29		begin local <block>]
qq 1.2 t a39	; 94 (takepoint2,takepoint2)		end else
qq 1.2 t a16	; 95 (takepoint2)		bypasslabel
qq 1.2 t a15	; 96 (takepoint1)		labeldecl
qq 1.2 t a15	; 97 (takepoint1)		begexpr
qq 1.2 t a15	; 98 (takepoint1)		begproc
qq 1.2 t a15	; 99 (takepoint1)		dolabel
qq t a38	;100 (unstack2, unstack2) if		
qq t a37	;101 (unstack2,unstack2,unstack1) release for		
qq 1.2 t a16	;102 (takepoint2)		forlabel
qqf 1.2+2.5+5 t a3+11d1.29	;103 hs c10 , qq e74+e73.29		↑real
hv r7.2+4.5+0 t a56	;104 (goto nexttop 2)		else
hv r 5 t a27 LT	;105 (goto top2 LT)		then
hv r 5 t a27 NT	;106 (goto top2 NT)		< then >then
hv r 5 t a27 LT	;107 (goto top2 LT)		≤ then ≥then
hv r 5 t a27 LZ	;108 (goto top2 LZ)		≠ then
hv r 5 t a27 NZ	;109 (goto top2 NZ)		= then
hv r 5 t a26 NT	;110 (goto top1 NT)		while... until...
hv r6.2+4.5+7 t a31	;111 (goto top1)		for...
hv r 5 t a41 LT	;112 (goto nexttop2 LT)		while..do until..do
hv r6.2+4.5+7 t a42	;113 (goto top2,unstack2)		goto bypass
d23:			
hv r6.2+4.5+7 t a40	;114 (goto nexttop 2)		for..do internal: else
qq 2.2+2.5+5 t a43	[115 qq 0, qq reltop1+tracktop1.29+Btop1.41		programpoint]

[Pass 8. page 16. Tables 5, finis tape pass 8]

```
[Finis tape pass 8]
```

```
d7: qq 0 , qq 0 ; start stack 1
d d8=1022 ; start stack 2
d d21=d8-d7-1 ; initial free stack
```

[Pass information for general pass and slip]

```
_e ; end pass 8 drumblock
```

 $|S|$

[27. dec. 1965]

[PASS 7]

b k =e15+e70,i=205e13,a30,b30,c90,d70

d e17=i

[The following are output byte values]

d d10=70 [clear], d11=27 [pm UA], d12=68 [clear R]
d d13=69 [clear RF], d14=1 [constant], d15=771[p rel]
d d16=772 [p rel], d17=40 [arnf], d18=41 [arn]
d d19= 3 [end exp], d20=47 [ps], d21=28 [gs UA]
d d22=45 [pm], d23=17 [end address expr],d24=75 [tk1]
d d25=259 [p rel], d26=258[abs adr], d27=5 [s rel]
d d29=5 [s rel], d30=2 [abs adr], d31=640[operand descr.]
d d32=-254[abs adr], d33=73 [end single do], d34=72 [end no]
d d35=120 [take for label],d37=121[take nonsense], d38=123[end pass]

[The following are input byte values]

d d28=80 [address], d36=91 [std.proc.no parameters]
d d39=39 [proc;]

[a-1] qq 6.9+e79.19-e82.19 ; operand for take value
a: qq d15 t 0 ; used in get next work
qq ; byte 1
qq ; byte 2
qq 512 t 768 ; operand Vin UA
qq 512 t 896 ; operand Vin UV
[5a] qq 256 ; used in get next w
qq 640 t 768 ; operand Ain UA
qq t 128 ;
qq 1 t 256 ; 2.0
[auxiliary stack] ;
qq d15 , ; constant term address
[10a] qq ; running lower bound
qq d15 , ; running coefficient 2
qq ; constant term (may be value)
qq d15 , ; length address
qq ; length (may be value)
[15a] qq d15 , ; running coefficient
qq t 256 ; 1.0
b5: qq ; work table
[18a] qq , ;
[19a] qq d15 , ; length position
[20a] qq d15 , ; array address
[21a] qq d15 , ; dope address
[22a] hs e5 ; error print on
[23a] qq , qq e55 ; overflow
[24a] hv 22a , hv 22a ; initiil alue for call 0

```

c74:arn(e1)   t 1  IZB      ; start pass 7:
hs  e2        LA         ;   initial stack reference:= input;
ga  b20      , pp  d8      ;   p:= bottom of operand stack;
pm  d38      DX          ;   output(endpass);
hs  e3        ;           ;   call (input );
hs  c21      , qq  a1      ;   prepare return to next;
grn  d8      t 1  M       ;   initialise operand stack;
it  (r-1)    , bs 39e25    ;
hv  r-2      ;
pm  24a      ;           ;   cell 0:= jump to error print
gm  0        MA          ;   on overflow;
hv  c51      ;           ;   goto blockstack;

```

[Central interpreter. The program instructions sit in the actionsstack, 4 instructions to a word. The instructions are in general interpreted in the following sequence:

8	9	10	11	stackbottom, a-marked
4	5	6	7	current action word
1	2	3		

Two special entries may change this sequence.

Skip: The first following action byte in the same word is skipped.

Next skip: The first following action byte is executed. All following actions in the same word are ignored.

The actionwords may be stacked in two ways. An inputbyte causing a look-up in the input control table and a storing of the table word in the stackbottom. This is the entry nextin. Actionwords are stacked explicitly in some actions.

The instructions may be interpreted in three ways. They are distinguished in the tables by using a b-, c- or d- name as base address and in the notes by using full underlining, no underlining or partial underlining.

Form	Value	Action
<u>b-name</u>	-256 to -1	Output(256 + b-name>)
<u>c-name</u>	0 to 511	Call action at address c + <c-name>
<u>d-name</u>	-512 to -257	Call output instruction (512 + <d-name>)

Output instruction will output the top operand followed by the instruction number (e.g. 512 + <d-name>). Finally the top operand is released from the operand stack.

Formats in operand stack.

	kind	operandbyte	part 2	part 3	class
bits:	0-2	3-9	10-19	20-29	40-41
V in R	100	0	0	0	R
V in RF	010	0	0	0	R
V in W	110	3.9	rel addr	0	no clear
A in W	110	4.9	rel addr	0	no clear
V abs	010	2.9	blockno	block0-rel	clear or no clear
V local	010	3.9	blockno	p-rel	clear or no clear
V block	000	5.9	blockno	s-rel	clear or no clear
V in UV	100	7.12	(0)	0	clear
V in UA	100	6.12	(0)	0	clear
take value	000	6.9	last used	s-rel	no clear
constant		1	bits 0.39 contain		constant
			the value		

class	:	R	constant	clear	no clear
bits 40-41	:	00	11	10	01

Storing of a complete operand is in the notes described in this way:
operand := <kind>,<part 2>,<part 3>,<class>;]

```

a1:          , ps  a1      ;   prepare return to next;
c:           ;
b1: pmn d2     XV   ITA    ; next: if current actionword = 0 then
[address of curr actword] ;   goto next actionword;
a2: pm 18a     V        ; after nextin:
    hv a3      LZ       ;   current action byte:=
    ga b2      , cl  10  ;   part1 (current actionword);
    gr (b1)    ;         shift current actionword 1 byte;
    qq        V      NZ  ;   if current actionword = 0
    qq (b1)    t -1 NA   ;   ^ address of curr actword ≠ bottom
                        ;   then address of curr actword
                        ;   := addr of curr actword - 1;
    grn(b1)    LZA      ;   if next skip then current actionword:=0;
b2: pmn 0      DXV 256 LTA ;   if current actionbyte ≥ 0 then
[current actionbyte] ;   switch to action[current actionbyte];
    hvn(b2)    t c      ;   if current actionbyte ≥ -256 then
    hv e3      NT       ;   begin output(current actionbyte + 256);
                        ;   goto next end;

c1: hs c5      ; output instruction: call (output operand);
    pm (b2)    D      256 ;   output (current actionbyte + 512);
    hs e3      X        ;
c27: pmn p     XV   NZB   ; release: if -, no release then
    arn a2     V     IZB   ;   begin p:= p - 1; if class[p+2] = clear
    pp p-1     ;   v class[p+1] = const then return;
    hr s1      LA        ;   end;
    pan b12    t d2 NC    ;   if class[p+1] = R v no release then
    hr s1      NT       ;   begin seet R not used;
                        ;   no release:= false; return end;
    gt r2      ;   if kind[p+1] = VinW v kind[p+1] = AinW
    pm 256     DX        ;   then
b3: ns 0       , ck  s 0   ;   release work location in
[wbase 1]      ;   worktable[part 2(operand[p+1])-wbase 1];
    sc b5      , hr  s1    ;   return;

a3: pmn(b1)    VX t-1 NA   ; next actionword:
                        ;   if address of curr actword ≠ bottom then
                        ;   begin address of curr actword:=
                        ;   address of curr actword - 1;
                        ;   next skip:= false; goto next end;
c2: arn(e1)    V 1  IOA    ; nextin:
    hv b1      IZA      ;
    hs e2      LA       ;   current actionword:=
    ga b       , ps  a1   ;   actiontable[input]; next skip:= false;
b:  pmn 0      Xt d1 ITA   ;   RC:= marks; goto after nextin;
[current inputbyte] ;   comment marks are used in +, -, ×, /
    hv a2      IRC       ;   and operands;

c3: pa (b1)    , hh  a1    ; skip: part1(current actionword):= =;
                        ;   goto next;

c4: hhn a1     IZA       ; next skip: next skip:= true; goto next;

```

```

c5: pmn p      X      ITA      ; output operand:
    hv a16     LC      ;      if class [p] = constant then goto outconst;
    tk 2       ITB      ;      if kind [p] = VinUVvkind [p] = VinUA
a10:cl 18      V      NTA      ;      v kind [p] = AinUA then goto out operand byte;
    hv a4      NTB      ;      if kind [p] = VinW v kind [p] = AinW then
    cl 8       LTA      ;      output (part 2)
a5:  hs e3      ;      else begin output (part 3);
    cln -10    NTA      ;      prepare output of part 2 end;
    hs e3      NTC      ;      if kind [p] = Vblock then output (part 2);
    cln -10    V        ;      prepare output of operand byte;
a4:  tk 1      ; out operand byte:
    hv e3      ;      output (operand byte); return;
a16:cl 30      X        ; outconst:
    ar d14     DX      ITC      ;      set constant operand byte;
    hs e3      ;      output (part 4); output (part 3);
    cln -10    ;      goto output remaining 3 bytes;
    hs e3      ;
    cln -10    , hv a5    ;
b12: ;
c6:  pm d2      ITA      ; clear R: R was used:= kind[R used in] = VinR;
    [R used in] ;
a7:  pmn b5     XV      NC      ;      if class[R used in] = R then
    hr s1      ;      begin
    hs a6      ;      call (get next work);
    hs e3      ;      output (w rel); operand[R used in] :=
    gm (b12)    MB      ;      VinW, w rel, 0, no clear;
    pm d12     DV      LTA      ;      if R was used then output (clear R)
    pm d13     D        ;      else output (clear RF);
    pa b12     t d2      ;      set R not used;
    hv e3      X        ;      end; return;

```

[The work table consists of a single machine word. Bit 0 is always 1. Bit n is 1 if working location n is occupied, 0 otherwise. The machine instruction nk with work table in the R-register will shift R until first free working location is in bit 1.]

```

c7:  pmn b5     X        ; get next work:
a6:  nk b13     , ar 5a   ;      scan work table for first free working loc;
b13:tk 0        , gr b5   ;      reserve work;
    [wrel]      ;
b14h:pm(b13)    DX 0      ;      Radr:= w rel:= w + wbase 2;
    [wbase 2]   ;
    it (b13)    , pt a     ;      form descr. of working loc;
    it (b13)    , bs (b6)  ;      if minimum w > w rel then
b6:  qq 0        t -1     ;      minimum w := minimum w - 1;
    [minimum w] ;
    pm a        , hr s1    ;      M:= descr of working loc; return;

```

[pass 7, page 6]

```

c8: pa  b2      V d17-256      ; top to RF release: current actionbyte:= arnf;
                                ; skip next line;
c9: pa  b2      t d18-256      ; top to R release: current actionbyte:= arn;
    pm  p                ; if class [p] = R then
    pa  b12     V d2  NC       ; begin set R not used; unstack; return end
    pm (b12)  V      ITA       ; else
    pp  p-1     , hr  s1       ;
    hs  a7      NC            ; call (clear R);
    hv  c1      ; goto output instruction;

c10:arn p      , sr  3a       ; top to UA rel: if kind [p] = VinUA then
    pa  b2      V d20-256NZ    ; begin unstack; return end;
    pp  p-1     , hr  s1       ; current actionbyte:= ps;
    hs  c1      ; call (output instruction);
    pm  d21     DX             ; output (gs UA);
    hv  e3      ; return;

c11:pm  256     DV            ; stack VinRF: Vin:= VinRF;
                                ; skip next line;
c12:pm  512     D             ; stack VinR: Vin:= VinR;
    gm  p1      M             ; operand [p+1]:= Vin, 0, 0, R;
    pp  p1      , gp b12      ; p:= R used in:= p + 1; return;
    hr  s1      ;
c13:pm  3a      V             ; stack VinUA: Vin:= VinUA; skip next line;
c14:pm  4a      ; stack VinUV: Vin:= VinUV;
    gm  p1      MA            ; operand [p+1] := Vin, 0, 0, clear;
    pp  p1      , hr  s1      ; p:= p + 1; return;

c15:pm  p      X      IQC     ; exchange: R:= operand [p];
    pm  p-1     ; M:= operand [p-1];
    gr  p-1     MQC          ; operand [p-1]:= R;
    qq (b12)  V 1  NC        ; if class [p-1] = R then
    qq (b12)  t -1 NQC       ; Rused in:= R used in + 1
    qq      IQC             ; else if class [p] = R then
    gm  p      MQC          ; R used in:= R used in - 1;
    hr  s1      ; operand [p]:= M; return;

```

```

c16:pa b10 , hv c19 ; clear UAUV: blocklimit:= 0; goto clear;
c17:pa b10 V 511 ; total clear: blocklimit:= big; goto clear;
c18:pa b10 t -128 ; clear UA: blocklimit:= part 2 (operand VinUV);
c19:pm (b12) ITA ; clear:
      hs a7 NC ; clear R if used;
      gp b9 ; i:= p;
      pmn p XV ITA ; get operand [p];
a8: ;
b9: pmn 0 Xt -1 ITA ; next clear: if class [i] = noclear
[running index for clear] ; v class [i] = constant then
      hv a8 LB ; begin i:= i - 1; goto next clear end;
      hr s1 NA ; if class [i] = R then return;
      gt b11 ;
b10: ;
b11:bs 0 t 0 ; if blocklimit ≤ part 2 (operand[i]) then
[blocklimit, blockno] ;
      tk 2 V ITB ; begin i:= i - 1; goto next clear end;
      hv a8 ; if kind [i] = VinUV v kind [i] = VinUA
      hv a9 LTA ; v kind [i] = AinUA then goto save UAUV;
      hs a10 ; call (output operand);
a11:pm d22 D ITA ; output save: addrop:= false;
a12:hs e3 X ; output (pm);
      hs c7 ; output store instr: call (get next work);
      hs e3 ; output (w rel);
      gm (b9) MB ; operand[i]:= VinW, wrel, 0, no clear;
      pa (b9) t d16 LTA ; if addrop then kind [i] := AinW;
      pm d10 D ; output (gm);
      hs e3 X ; i:= i - 1; goto next clear;
      hv a8 ;
; save UAUV: addrop:= kind [i] = AinUA;
a9: tk 1 ITA ; if addrop then
      hs e3 NTA ; begin output (operand byte[i]);
      hv a11 NTA ; goto output save end;
      pm d11 D ; output (pm UA);
      hv a12 ; goto output store instr;

c20:pmn(e1) X 1 ; input 2: byte 2:= input;
      hs e2 LA ;
      ga 2a ;
c21:pmn(e1) X 1 ; input 1: byte 1:= input; return;
      hs e2 LA ;
      ga 1a , hr s1 ;

```

```

c22:pmn(e1)    X 1          ; inout 4: output (input);
      hs  e2          LA    ;
      hs  e3          ;
c23:pmn(e1)    X 1          ; inout 3: output (input);
      hs  e2          LA    ;
      hs  e3          ;
c24:pmn(e1)    X 1          ; inout 2: output (input);
      hs  e2          LA    ;
      hs  e3          ;
c25:pmn(e1)    X 1          ; inout 1: output (input);
      hs  e2          LA    ;
      hv  e3          ;    return;

c26:pm  p      X          ; prepare assign: if class [p] = R then
      pa  b12    V d2 NC   ;    set R not used else
      hv  a13    NZ       ;    if operand [p]  $\neq$  0 then goto assign var;
      pa  b4     V -10LZ   ;    if operand [p] = 0 then assignkind := grn else
      pa  b4     V 10 LT   ;    if kind [p] = VinR then assignkind := gr else
      pa  b4     NZ       ;    assignkind:= grf;
      pp  p-1    , hr  s1   ;    p:= p - 1; return;
a13:pa  b4      t 20       ; assign var: assignkind:= gm;
      pa  b2     t d22-256 ;    current actionbyte:= pm;
      hv  c1     ;    goto output instruction;

[c27: release, see page 2];

c28:pm  p-1    X          ; two operands: not used;
      hv  c8     NA       ;    if class [p-1] = clear
      nc  d27    NB       ;    ^ kind [p-1] = Vblock
      hv  c8     ;    then begin
      pp  p-1    ;    p:= p - 1;
      hs  c8     ;    call (top to RF release);
      hs  c11    ;    call (stack VinRF); p:= p + 1 end;
      pp  p1     , hv  c8  ;    goto top to RF release;

c29:pm  p      ; to cell and RF:
      hv  c6     NC       ;    if class [p] = R then goto clear R;
      hs  c6     ;    call (clear R);
      pa  b2     t d17-256 ;    current actionbyte:= arnf;
      hvn c1     IZB      ;    no release:= true; goto output instruction;

```

```

c30:pm (b8)    D          ; array:
      gm p1      MC      ; operand [p+1]:= subcount, 0, 0, const;
      pp p1      ;       p:= p + 1;
      pa b8      t -1    ;       subcount:= -1;

c31:arn 2a      , ck 20   ; operand: stackrel [p+1]:= byte 2;
      qq         V      LRB ; if operand  $\neq$  variable  $\wedge$  operand  $\neq$  own then
      gr p1      V      MB  ; class [p+1] := no clear else
      gr p1      MA      ; class [p+1] := clear;
      pm 1a      X       ; if byte 1 = current blocknumber
b7: ca -1       V      NRA  ;  $\wedge$  operand  $\neq$  own then
      [current blocknumber] ;
      qqn        V      ; kind [p+1]:= Vlocal else
      pan p1     V d25    ; if byte 1 = 0  $\vee$  operand = own then
      pa p1      V d26 LZ ; kind [p+1] = Vabs else
      pa p1      t d27 NZ ; kind [p+1]= Vblock;
      it (1a)    , pt p1  ; blockno [p+1]:= byte 1;
      pp p1      , hv c2  ; p:= p + 1; goto nextin;

c32:pmn(e1)    X 1       ; procedure:
      hs e2      LA      ; if input address then
      nc d28     , hh a14 ; begin
      qq (1a)    t 1      ; byte 1:= byte 1 + 1;
      pa 2a      t -1     ; byte 2:= -1; goto operand;
a14:hv c31      , is (1a) ; end change operand to value funct.des;
      it s1      , pa b10 ; blocklimit:= byte 1 + 1;
      qq (e1)    V -1     ; save inputbyte; skip next line;
c33:it (1a)     , pa b10  ; formal: blocklimit:= byte 1
      hs c19      ; call (clear);
      hs c13      ; call (stack VinUA);
      pm 2a      X       ;
      hs e3      ; output (byte 2)
      pm 1a      X       ; if byte 1 = current blocknumber then
      ca (b7)    , hv c   ; goto next;
      hv a15     LZ      ; if byte 1  $\neq$  0 then
      pm 1a      ; begin
      hs e3      X       ; output (blocknumber);
      pm d29     DVX     ; output (s rel)
a15:pm d30      DX      ; end else output (abs adr);
      pa (b1)    , hv e3  ; skip next action; return;

c34:pan r3      X 3       ; constant:
      arn(e1)    t 1      ; p:= p + 1;
      hs e2      LA      ; operand [o]:=
      bt 0       t -1     ; input, input, input, input, constant;
      cl 10      , hv r-3 ; goto nextin;
      cl -30     , pp p1  ;
      gr p       MC      ;
      hv c2      ;

```

```

c35:pm  p          ITA      ; address: if kind [p] = VinUA then
      pa  p          V d31LTA ;      kind [p]:= AinUA else
      gm  p          MB      ;      class [p]:= no clear;
      hv  c          ;      goto next;

c36:pm  (e1)      X          ; parameter: if blockno = 0 then
      hv  c4          LZ      ;      goto next skip;
      hv  c3          ;      goto skip;

c37:arn  p          ; relation: if operand [p] = 0 then
      pm  842      DV  LZ      ;      begin a:= 6; b:= 4; c:= 5 end else
      pm  176      DV  NC      ;      if class [p] = R then
                                   ;      begin a:= 1; b:= 3; c:= 0 end
      pm  320      D    NZ      ;      else begin a:= 2; b:= 4; c:= 0 end;
      arn  p-1      X    IZB      ;      if operand [p-1] = 0 then
      tk  3          V    LZB      ;      i:= b else
      tk  6          NC      ;      if class [p-1] = R then i:= c
      ck  -7          , ga  r4      ;      else i:= a;
      arn(b1)        , ga  r4      ;      i:= i + next actionbyte;
      tk  10          , gr  (b1)    ;      skip next actionbyte;
      qq  (b1)      t 1  IZB      ;      stack new action (i + relation table base);
      is  0          ;      goto next;
      pmns0          Xt d3 ITA      ;
      hv  a2          IRC      ;

c38:pm  p-1          ; boolean op: if class [p-1] = R then
      hv  c15          NC      ;      goto exchange;
      hr  s1          ;      return;

c39:pm  p-1          ; plus times: if class [p-1] = R then
      hs  c15          NC      ;      call (exchange);
c40:pm  p-1          ; minus divide:
      arnf p          LC      ;      if class [p-1]  $\neq$  R  $\vee$  class [p]  $\neq$  R then
      pp  p-1      V    LC      ;      goto top to RF release;
      hv  c8          ;      p:= p - 1;
      arf  p          V    NRC      ;      operand [p]:=
                                   ;      (if add then operand [p+1] + operand[p]
      srf  p          V    NRA      ;      else if sub then operand [p+1] - operand [p]
      mkf  p          V    LRC      ;      else if mult then operand [p+1]×operand [p]
      dkf  p          LRA      ;      else operand [p+1]/operand [p]);
      grf  p          ;      clear remaining actionword;
      grn(b1)        , hr  s1      ;      return;

```



```

c41:arn p          IQC      ; endexpr: remove shield;
    pp p-1          ;      p:= p - 1;
    gr p            MQC      ;      if class [p] = R then
    qqn(b12) V -1 NC      ;      begin R used in:= R used in - 1:
                                ;      goto skip end;
    hv c3           LC      ;      if class [p] = constant v kind [p] = Vabs
    hv c3           NT      ;      v kind [p] = Vlocal v kind [p] = Vblock then
                                ;      goto skip;
    pp p-1          , sr 4a   ;      p:= p - 1;
    hv c4           LZ      ;      if kind [p] = VinUV then goto next skip;
    pm d23          DX      ;      output (end address expr);
    ps c2-1         , hv e3   ;      goto nextin;

c42:pm p           X        ; goto: if kind [p] = VinUA then
    ck 10           V      NT ;      begin p:= p - 1; goto next skip end;
    pp p-1          , hv c4   ;      if part2 (operand[p]) = current blockno then
    ca (b7)         , hv c3   ;      goto skip;
    pm d40          , gm (b1) ;      current actionword:= non local goto;
    hv c            ;      goto next;

c43:qqn            IZB      ; while assign: no release:= true;
c44:pmn d41        XV      ; colon equal: assign word:= grf, gr, grn;
                                ;      skip next line;
c45:pmn d42        X      IZB ; for assign: assign word:= grf M, gm M, grn M;
b4: ck 0           , ga b2   ;      no release:= true;
    [assign kind]      ;      current actionbyte:=
                                ;      byte (assignkind, assign word);
    hv c1            ;      goto output instruction;

c46:pm d43          ; end subscripts: stack new action (top to RF rel.
    gm (b1) t 1      ;      round, indexcall);
c47:pm d5           IRC      ; subscr comma:
    arn (b8) D       ;      if subcount ≥ 0 then
    gm (b1) t 1 NT    ;      stack new action (+);
    hr s1            ;      return;

c48:hs c8           ; not last subscript: call (top to RF release);
b8: pm 0           DXt 1     ;      subcount:= subcount + 1;
    [subcount]      ;
    ar d44          V      NZ ;      if subcount = 0 then
    hvn c           IZB      ;      begin no release:= true; goto next;
                                ;      comment remaining actions:
                                ;      mkf(). stackV in RF; end;
    ck -10          , gr (b1) ;      current action word:=
                                ;      ps(), indexmult, stack V in RF;
    hvn c           IZB      ;      no release:= true; goto next;

c49:it (p)         , pa b8   ; last subscript: subcount:= part 1 (operand[p]);
    pm 3a           ;      operand [p] := VinUA, 0, 0, clear;
    gm p            MA      ;      goto nextin;
    hv c2           ;

```

```

; entier: current actionword:=
c50:pm d6      , gm (b1) ; top to RF rel, stack VinRF, round, float;
    hv c3      ; goto skip;

c51:pm b5      , gm p1   ; blockstack: stack work table;
    pm (b6)    D        ; operand [p+2]:=
    gm p2      M        ; minimum working, wbase 2, 0, R;
    arn b14    , gt p2   ; comment shield for clear;
    it (1a)    , pa b6   ; minimum working:= byte 1;
    it (1a)    t -1     ; wbase:= byte 1 - 1;
    pa b3      , pp p2   ; p:= p + 2;
    it (1a)    , pt b14  ; wbase 2:= byte 1 - 1;
    pm 512     D        ; work table := no reserved;
    gm b5      , hr s1   ; return;

c52:qq (b7)    t -1     ; blockunstack: currentblockno:= currentblockno - 1;
    pm (b6)    D -2     ; output (minimum working - 2);
    arn p      , ga b6   ; minimum working := part 1 (operand[p]);
    gt b14     , ck 10   ; wbase 2:= base 1:= part 2 (operand[p]);
    ga b3      , arn p-1 ; work table:= operand [p-1];
    gr b5      , pp p-2  ; p:= p - 2;
    hv e3      X        ; return;

c53:pm (b7)    D 1      ; beglocal: current blockno:= currentblockno + 1;
    hv e3      X        ; output(current blockno); return;

c54:pm d33     D        ; for: operand [p+1]:= end single do, 0, 0, R;
    gm p1      M        ;
    hs c7      ; call (get next work);
    gm p2      MB       ; operand [p+2]:= VinW, wrel, 0, no clear;
    pp p2      , hr s1   ; p:= p + 2; return;

c55:arn p-1    , tk 10   ; get forlabel:
    hs e3      ; output (wrel in operand[p-1]);
    pm d35     DX       ; output (take for label);
    hv e3      ; return;

c56:hs c29     ; until: call (to cell and RF); comment step to RF;
    pmn p-1     IZB     ; no release:= true;
    gm p1      MB       ; operand [p+1] := operand [p+2]
    gm p2      MA       ; := operand [p+3]:= operand [p-1];
    gm p3      MA       ; comment copy descr. of controlled variable.
    pp p3      , hr s1   ; Used by actions: arf, grf VLA, acn MA;
    ; p:= p + 3; return;

```

```

; stepelement: comment operand stack contains
;   p - 5 : end do
;   p - 4: work, for label
;   p - 3: controlled, simple or AinW
;   p - 2: step value
;   p - 1: controlled, copy of p - 3
;   p:      until value;
c57:acn p-1      MA      ;   class [p-1]:= clear;
      arnfp-2      ;   if class [p-2] = const then
      pm d45      V      LC      ;   stack new action (top to RF rel,
      ;   srf, release) else
      pmn d46      ;   stack new action (top to RF rel, srf, mkf);
      gm (b1)      t 1      ;   if class [p-2] = const ^ operand [p-2] < 0 then
      hv c15      LT      ;   goto exchange;
      hr s1      ;   return;

c58:pa p-2      t d34      ; set more for: replace end single do by end do;
      hr s1      ;   return;

c59:pm d47      ; simple for: stack new action (while assign,
      gm (b1)      t 1      ;   get forlabel, doabs, forlabel);
      hv c26      ;   goto prepare assign;

c60:arn p1      , tk 10      ; enddo: output (wrel in operand[p+1]);
      hs e3      ;   comment contains forlabel;
      pp p-1      , pm p1      ;   output (enddo or end single do);
      hv e3      X      ;   p:= p - 1; return;

c61:pm d59      , ud a27      ; real exp: stack new action(stack VinUV,
      ;   arnf, mkf, exp);
      pm d58      ;   stack new action (clear UAUUV,
a27:gm (b1)      t 1      ;   exchange, top to RF rel, areal);
      hr s1      ;   return;

c62:pmn p      X      ; integexp: if class[p] = const
      sr 8a      LC      ;   ^ (operand[p] = 2.0 v operand[p] = 3.0) then
      sr 7a      NZ      ;   begin
      hv a28      NZ      ;       p:= p - 1;
      pp p-1      ;       call (to cell and RF);
      hs c29      ;       current action word:=
      pm d50      , gm (b1) ;       mkf, mkf, stack VinRF;
      arn p1      , sr 8a      ;       if operand[p+1]: 2.0 then
      hv c3      LZ      ;       goto skip;
      hvn c      IZB      ;       no release:= true; goto next
a28:pm p      X      ;   end;
      pm d62      V      NC      ;   if class[p] = R then
      pm p-1      XV      ;   begin stack new action (grf c68,
      hv a27      ;   release, pm, gmc54); return; end;
      pm d63      V      NC      ;   if class [p-1] = R then
      qq      V      ;   begin stack new action (grf c54,
      hv a27      ;   pm, gmc68, release); return; end;
      hs c6      ;   call (clear R);
      pm d64      , hv a27      ;   stack new action (pm, gmc68, pm, gmc54);
      ;   return;

```

```

c63:it (1a)      , pa b15      ; bounds: number of arrays:= byte 1;
      it (2a)      , pt  9a      ;   constant term address:= VinW, byte 2, 0, clear;
      it (2a)      t 1          ;   running coefficient 1:=
      pt 15a      , it 1          ;   VinW, byte 2 + 1, 0, clear;
      it (2a)      , pa  b16      ;   final coef address:= byte 2 + 2;
      pa b17      t 2          ;   bound type:= 2;
      pm 15a      , gm  11a      ;   running coefficient 2:= length address
      gm 13a      , pa  b18      ;   := VinW, byte 2 + 1, 0, clear;
      hv c2                ;   still constant:= true; goto nextin;

c64:arnf p-1                ; boundcomma: if lower bound  $\neq$  const then
      hv a17                ;   begin p:= p + 1;
      pp p1      , pm 16a      ;   operand[p]:= 1.0, const;
      gm p                ;   stack new action (top to RF rel,
      pm d51                ;   no release, srf, stack Vin RF);
      gm (b1)      t 1          ;   stack new action (plustimes, arf,
      pm d5      , hv  a18      ;   stack VinRF); return;
a17:srf 16a                ;   end;
      grf p1                ;   operand[p+1]:= lower bound - 1.0, const;
      hr s1                ;   if lower bound = 1.0 then return;
      pp p1      , pm  d7      ;   p:= p + 1; stack new action (exchange,
a18:gm (b1)      t 1  IRC      ;   minus divide, srf, stack VinRF);
      hr s1                ;   return

c65:qq (b17)  t 1          ; last bound: bound type:= bound type + 1;
                                ;   comment operand stack contains:
                                ;   p - 1: li = lower bound
                                ;   p      : ci: uppwer - lower + 1;

c66:pm p-1          IRC      ; not last bound:
      gm 10a          MRC      ;   running lower bound:= operand [p-1];
      gp b19      , pm  p      ;   save top:= p;
b18:can 0          V  LC      ;   if still constant  $\wedge$  class [p] = const then
      [still constant]      ;
      pa b18      V 1          ;   switch to proper bound type action;
      is (b17)      , hh  sa19  ;   still constant := false;
b17:pm 0          t d53      ;
      [bound type]          ;   current actionword:=
      gm (b1)                ;   boundword [boundtype];
      pm (b17)  t 4          ;   boundtype:= boundtype + 4;
      gm (b1)      t 1          ;   stack new action (boundword [boundtype]);
      hs c8                ;   call (top to RF rel);
      pp 15a      , hr  s1      ;   p:= address of running coef1; return;
                                ;   comment the new actions will work on
                                ;   the aux. stack generating part of code
                                ;   for length and constant term. The
                                ;   aux. stack is described on 1 pass 7;

```

```

c67:pm 13a          ; reset: length:= length address;
    gm 14a          MA ; constant term:= const term address;
    pm 9a           ; comment possibly a const. description is
    gm 12a          MA ; replaced by VinW;
c69:                ;
b19:pp 0           Vt -2 ; after bounds: p:= save top - 2;
    [save top]      ;
    qq t 1          ; increase address of running coef 1;
    arn r-1 , ac 15a ; increase address of running coef 2;
    ac 11a , pa b17 ; boundtype:= 0; goto nextin;

a19h:hv c2 , arnf 14a ; entries to constant boundtype actions:
    hv a21 , arnf 14a ; goto constant not first not last;
    hv a22 , pm p-1   ; goto constant not first last;
    hv a26 , arnf p    ; goto constant first not last;

    tkf -29 , pp 14a   ; constant first last: length:= round(ci);
    gr 14a , MC        ; p:= address of length;
    arnf10a , tkf -29  ; constant term:=
    gr 12a , MC        ; round (running lower bound);
    pm d52 , hv a23    ; stack new action (pm, gm,
    ; pm, count p2); goto next;

a21:mkf p , grf 14a   ; constant not first not last: length:= length×ci;
    arnf12a , mkf p    ; constant term:=
    arf p-1 ,          ; constant term×ci + li;
    grf 12a , MC       ; goto store ci;
    hv a24            ;

a22:mkf p , tkf -29   ; constant not first last:
    gr 14a , MC        ; length:= round (length×ci);
    arnf12a , mkf p    ; constant term:=
    arf p-1 , tkf -29  ; round (constant term×ci + li);
    gr 12a , MC        ; stack new action (pm, gm, pm, count p2);
    pm d52 , ud r3     ; output store ci:
a24:hs c5            ; call (output operand);
    pp 15a , pm d57    ; comment outputs ci;
a23:gm (b1) t 1      ; stack new action (pm, gm);
    hv c              ; p:= address of running coef1; goto next;

a26:gm 12a          MC ; constant first not last: constant term:= li;
    pm p             ; length:= ci;
    gm 14a          MC ; goto after bounds;
    hv c69           ;
s                ;

```

```

c70:pm d54 , gm (b1) ; end decl: current action word:=
    hv c ; round, gr, end const decl;
    ; goto next;

c68:pm 13a , gm 19a ; end const decl: length position:= length address;
b16:it 0 , pt 21a ; dope address:=
    [final coef address] ; VinW, final coef addr, 0, clear;
    qq (r-1) t -2 ; final coef addr:= final coef addr - 2;

c76:it (r-2) t -1 ; decl array: final coef addr:= final coef addr - 1;
    pt 20a , it -1 ; array address:=
b15:bt 0 , hv a25 ; VinW, final coef addr, 0, clear;
    [number of arrays] ; number of arrays:= number of arrays - 1;
    pp (b19) t -2 ; if number of arrays < 0 then
    hv c2 ; begin p:= save top - 2; goto nextin end;
a25:pm d56 , gm (b1) ; current actionword:= qq-lm hvs2, decl array;
    pp 21a , pm d55 ; stack new action (take last used,
    gm (b1) t 1 ; ar_D, gr_M, reserve space);
    hv c ; p:= address of dope address;
    ; goto next;
    ; comment the actions will work on dope
    ; address, array address and length position;

c77:pp p-2 , hr s1 ; count p2: p:= p - 2; return;

c71:pp p-1 , hr s1 ; count p: p:= p-1; return;

c72:hrn s1 IZB ; no release: no release:= true; return;

c78:grn p1 M ; shield: operand[p+1]:= 0, 0, 0, R;
    pp p1 , hr s1 ; p:= p + 1; return;
    ; comment shield for later clear actions;

c79:pm (e1) X 1 ; test for procedure:
    hs e2 LA ; if nextin = proc; then
    ca d39 , hv c ; goto next;
    pm d37 D ; output (take nonsense);
    hs e3 X ; save inputbyte; goto stack VinRF;
    qq (e1) V -1 ;

c80:hs c8 ; trykproc 1 par: call (top to RF rel);
    hv c11 ; goto stack VinRF; comment not used;
b20:
c73:pm 0 DX ; end pass 7:
[initial stackreference] ;
    hs e3 ; output (initial stack reference);
    pm 40e25 X -1 ; scan operand stack
    hv r2 NZ ; for first not used location;
    hv r-2 NC ;
    nt d8 , it (r-3) ; pass inf 1:= max depch of stack;
    pa 2e4 , hv e92 ; goto endpass;

```

[10.1.66]

[pass 7, page 17]

```
c75:arn(e1)      , hv  e3      ; last out: not used;

c81:pm  p        , ps  c-1     ; jump on boolean: if class[top] = R then
      pm  d24     VDX  NC      ;      output(tk1);
      hv  c        ;      goto next;
      hv  e3        ;

c82:pm  d66      , ud  a27     ; take value rounded: stack new action (formal,
      ;      arnf(UA), round);

c83:hs  c21      ; take value: call (input 1);
      ck  20      , ar  a-1    ;      operand [p+1] := operand [p+2]:=
      gr  p1      MB        ;      Vblock+AinUA, last used,
      gr  p2      MB        ;      blockrelative, no clear;
      pp  p2      , hv  c      ;      p:= p + 2; goto next;

c84:pm  p        IQC        ; array in normal mode:
      gm  p-1      MQC        ;      remove shield; p:= p - 1;
      pm  d9      , pp  p-1    ;      current actionword:=
      gm  (b1)    , hv  c      ;      top to R rel, inout 2, call st proc 1Rparam,
      ;      stack VinUV; goto next;

c85:pm  d68      , hv  a27     ; integer divide: stack new action (dkf,
      ;      srf c35VNT, arf c35, tkf-29,nkf 39);
      ;      return;

d2:qq      ,              ; bottom of action stack.
      qq          ;
      qq          ;
      qq          ;
      qq          ;
```

[Prepare symbolic names used in tables]

d d=512[base for d-operations], b=768 [base for b-operations]

[The following definitions for address relative to c]

```
d          c3 =c3 -c
d c6 =c6 -c, c7 =c7 -c, c8 =c8 -c, c9 =c9 -c, c10=c10-c
d c11=c11-c, c12=c12-c, c13=c13-c, c14=c14-c, c15=c15-c
d c16=c16-c, c17=c17-c, c18=c18-c, c19=c19-c, c20=c20-c
d c21=c21-c, c22=c22-c, c23=c23-c, c24=c24-c, c25=c25-c
d c26=c26-c, c27=c27-c, c28=c28-c, c29=c29-c, c30=c30-c
d c31=c31-c, c32=c32-c, c33=c33-c, c34=c34-c, c35=c35-c
d c36=c36-c, c37=c37-c, c38=c38-c, c39=c39-c, c40=c40-c
d c41=c41-c, c42=c42-c, c43=c43-c, c44=c44-c, c45=c45-c
d c46=c46-c, c47=c47-c, c48=c48-c, c49=c49-c, c50=c50-c
d c51=c51-c, c52=c52-c, c53=c53-c, c54=c54-c, c55=c55-c
d c56=c56-c, c57=c57-c, c58=c58-c, c59=c59-c, c60=c60-c
d c61=c61-c, c62=c62-c, c63=c63-c, c64=c64-c, c65=c65-c
d c66=c66-c, c67=c67-c, c68=c68-c, c69=c69-c, c70=c70-c
d c71=c71-c, c72=c72-c, c73=c73-c,          c75=c75-c
d c76=c76-c, c77=c77-c, c78=c78-c, c79=c79-c, c80=c80-c
d c81=c81-c, c82=c82-c, c83=c83-c, c84=c84-c, c85=c85-c
```

The notes conform to the following syntax:

```

d1: qq c53.9+ 93b.19 ; 0 beglocal; beglocal, beglocal
qq l18b.9 ; 1 endllocal; endllocal
qq c17.9+ c25.19+ 78b.29 [ 2 begcall; totalclear, inout 1,
begcall]
qq c21.9+ 98b.19+ c51.29 [ 3 begproc; input 1, begproc,
blockstack]
qq c25.9+129b.19+ c52.29+ 90b.39 [ 4 endprocedure; inout 1, ps_p,
blockunstack, endproc]
qq c25.9+129b.19+ c52.29+ 91b.39 [ 5 endtypeprocedure; inout 1, ps_p,
blockunstack, endtypeproc]
qq c21.9+ c51.19 ; 6 begblock; input 1, blockstack
qq c52.9+ 92b.19 ; 7 begblock; blockunstack, endblock
qq l13b.9 ; 8 gotobypasslabel; goto bypasslabel
qq 96b.9 ; 9 labeldeclaration; labeldeclaration
qq c55.9+102b.19 ; 10 whilelabel; getforlabel, forlabel
qq c78.9+ 97b.19 ; 11 begex; shield, begex
qq c17.9+100b.19 ; 12 if; totalclear, if
qq l04b.9 ; 13 else statement; else
qq 94b.9 ; 14 end else statement; endelse
qq c54.9+101b.19 ; 15 for; for, releasefor
qq c35.9+ c18.19 ; 16 := for; address, clear UA
qq c59.9+ c58.19 ; 17 simplefor; simplefor, setmorefor
qq c59.9+114b.19+ 99b.29 [ 18 simplefor and do; simplefor,
bypassabs, dolabel]
qq c26.9+ c43.19 ; 19 while; prepareassign, whileassign
qq c81.9+ c9.19+110b.29+ c58.39 [ 20 whileelement; jump on Boolean, top
to R rel, do NT, setmorefor]
qq c81.9+ c9.19+112b.29+ 99b.39 [ 21 whileelement and do; jump on Boolean,
top to R rel, bypass LT, dolabel]
qq c26.9+ c45.19+ c55.29+102b.39 [ 22 step; prepare assign, for assign,
get forlabel, forlabel]
qq c56.9+ 31d.19+ 48d.29+ 49d.39 ; 23 until; until, arf, grf VLA, acn MA
qq c57.9+110b.19+ c58.29 [ 24 stepelement; stepelement, do NT,
setmorefor]
qq c57.9+112b.19+ 99b.29 [ 25 stepelement and do; stepelement,
bypass LT, dolabel]
qq c27.9+ c27.19+ c60.29+ 95b.39 [ 26 enddo; release, release, enddo,
bypasslabel]
qq c46.9+ c49.19 [ 27 endsubscripts; endsubscripts, last
subscript]
qq c47.9+ c48.19+ 54d.29+ c11.39 [ 28 subscriptcomma; subscriptcomma, not
last subscript, mkf(), stack V in RF]
qq c8.9+ 14b.19+ 60d.29+ c13.39 [ 29 endswitchdes; top to RF rel, round,
switchcall, stack V in UA]
d6: qq c8.9+ c11.19+ 14b.29+ 23b.39 [ 30 round; top to RF rel, stack V in
RF, round, float]

```



```

qq c6.9+ 29d.19+ c11.29 ; 31 abs; clear R , annf, stack V in RF
qq c17.9+ c84.19 [ 32 std.proc. 1 array par; total clear,
array normal]
qq c8.9+ 22b.19+ c50.29 [ 33 entier; top to RF rel. srf half,
entier]
qq c9.9+ c12.19+ 18b.29 [ 34 -,; top to R rel, stack V in R,
ab 0 DX]
qq c6.9+ 30d.19+ c11.29 ; 35 neg; clear R , srnf, stack V in RF
qq c41.9+ 13b.19+ c8.29+ 15b.39 [ 36 end RF expr; endexpr, endexpr, top
to RF rel, end RF expr]
qq c41.9+ 13b.19+ c9.29+ 16b.39 [ 37 end R expr; endexpr, endexpr, top
to R rel, end R expr]
qq c10.9+ 17b.19+ c71.29 [ 38 endadresseexpr; top to UA rel,
endadresseexpr, count P]
qq c27.9 ; 39 proc;; release
qq c8.9+104b.19 ; 40 else RF expr; top to RF rel, else
qq c9.9+104b.19 ; 41 else R expr; top to R rel, else
qq c10.9+104b.19 [ 42 else address expr; top to UA rel,
else]
qq c8.9+ 94b.19+ c11.29 [ 43 end else RF expr; top to RF rel,
endelse, stack V in RF]
qq c9.9+ 94b.19+ c12.29 [ 44 end else R expr; top to R rel,
endelse, stack VinR]
qq c10.9+ 94b.19+ c13.29 [ 45 end else address expr; top to UA rel,
endelse, stack V in UA]
qq c81.9+ c9.19+105b.29 [ 46 then; jump on Boolean, top to R rel,
then]
qq c26.9 [ 47 prepare assign arith;
prepare assign]
qq c26.9 ; 48 prepare assign bool; prepare assign
qq c44.9 ; 49 :=; colon equal
qq c42.9+ 25b.19+ 47d.29+ 26b.39 [ 50 goto; goto, gotocomputed, ps,
gotolocal]
d5: qq c39.9+ 31d.19+ c11.29 ; 51 +; plustimes, arf, stack V in RF
d7: qqfc15.9+ c40.19+ 36d.29+ c11.39 [ 52 -; exchange, minus divide, srf,
stack V in RF]
qqfc39.9+ 32d.19+ c11.29, ; 53 x; plustimes, mkf, stack V in RF
qq c15.9+ c40.19+ 37d.29+ c11.39, [ 54 /; exchange, minus divide, dkf,
stack V in RF]
qq c15.9+ c8.19+ c85.29+ c11.39 [ 55 :; exchange, top to RF rel, integ
div, stack V in RF]
qq c62.9+130b.19+ c11.29 ; 56 int; integ exp, integ, stack V in RF
qq c61.9+ c14.19 ; 57 real; real exp, stack V in UV
qq c37.9+ 106b.29 ; 58 then; relation, hop NT
qq c37.9+ 7.19+107b.29 ; 59 ≤ then; relation, hop LT
qq c37.9+ 14.19+109b.29 ; 60 = then; relation, hop NZ
qq c37.9+ 107b.29 ; 61 ≥ then; relation, hop LT
qq c37.9+ 7.19+106b.29 ; 62 > then; relation, hop NT
qq c37.9+ 14.19+108b.29 ; 63 ≠ then; relation, hop LZ
qq c37.9+ 7.19+ 19b.29+ c12.39 ; 64 <; relation, sreps, stack V in R
qq c37.9+ 7.19+ c12.29 ; 65 ≤; relation, stack V in R
qq c37.9+ 21.19+ 20b.29+ c12.39 [ 66 =; relation, arn 512 DNZ, stack
V IN R]
qq c37.9+ c12.29 ; 67 ≥; relation, stack V in R
qq c37.9+ 19b.29+ c12.39 ; 68 >; relation, sreps, stack V in R
qq c37.9+ 28.19+ 19b.29+ c12.39 ; 69 ≠; relation, sreps, stack V in R

```

```

qq c38.9+ c9.19+ 33d.29+ c12.39 [ 70 λ; boolean, top to R rel, ab,
stack V in R]
qq c38.9+ c9.19+ 34d.29+ c12.39 [ 71 v; boolean, top to R rel, mb,
stack V in R]
qq c38.9+ c9.19+ 35d.29+ c12.39 [ 72 ≡; boolean, top to R rel, mbX,
stack V in R]
qq c18.9+ c20.19+ c30.29 ; 73 array; clear UA, input 2, array
qqfc20.9+ c31.19 ; 74 variable; input 2, operand
qqfc20.9+ c31.19, ; 75 own; input 2, operand
qq c20.9+ c31.19 ; 76 switch; input 2, operand
qq c20.9+ c31.19 ; 77 label; input 2, operand
qq c20.9+ c33.19+ 3b.29+ 55b.39 [ 78 formal; input 2, formal, p_rel,
formal]
qq c20.9+ c32.19+ 3b.29+ 55b.39 [ 79 procedure; input 2, procedure,
p_rel, formal]
qq c35.9 ; 80 address; address
qq c25.9+ 76b.19 ; 81 localswitch; inout 1, localswitch
qq c21.9+115b.19 [ 82 local procedure; input,
programpoint]
qq115b.9 ; 83 local label; programpoint
qq 12b.9 ; 84 blind; blind
qq115b.9 ; 85 expression; programpoint
qq c24.9+ c36.19+ 82b.29+ 81b.39 [ 86 simple or label; inout 2, parameter,
simple block 0, simple]
qq c24.9+ 82b.19 ; 87 own param; inout 2, simple block 0
qq c24.9+ c36.19+ 84b.29+ 83b.39 [ 88 described in stack; inout 2, parameter,
descr in stack block 0,
described in stack]
qq c24.9+ 80b.19 ; 89 std.proc.par; inout 2, standardproc
qq c22.9+ 77b.19 ; 90 constant par; inout 4, parameterconst
qq c16.9+ c24.19+ 87b.29+ c14.39 [ 91 st.proc. no par; clear UAUV, inout 2,
call st proc no par, stack V in UV]
qq c8.9+ c24.19+ 88b.29+ c14.39 [ 92 st.proc.1 par; top to RF rel, inout 2,
call st proc 1 par, stack V in UV]
d9: qq c9.9+ c24.19+ 89b.29+ c14.39 [ 93 st.proc. 1R par; top to R rel,
inout 2, call st proc 1 R par,
stack V in UV]
qq c16.9 ; 94 prepfunc; clear UAUV
qq c34.9 ; 95 constant; constant
qq119b.9+ c24.19+ 85b.29+ c14.39 [ 96 end proc. call; return inform,
inout 2, endcall, stack V in UV]
qq119b.9+ c24.19+ 79b.29+ c14.39 [ 97 end st.call; return inform, inout 2,
end call st., stack V in UV]
qq c20.9+ c63.19 ; 98 bounds, input 2, bounds
qq c64.9+ c66.19+ c69.29 [ 99 boundcomma; boundcomma, not last
bound, after bounds]
qq c64.9+ c65.19+ 46d.29+ c68.39 [100 end bounds; boundcomma, last bound,
gm, end const decl]
qq c82.9+ 23b.19+ 42d.29 [101 take value rounded; take value
rounded, float, grf]
qq c83.9+ 55d.19+ 65b.29+ 46d.39 [102 take value unrounded; take value,
formal, pm(UA), gm]
qq c73.9 ;103 endpass; endpass
qq 95b.9 ;104 bypasslabel; bypasslabel
qq ;105 take nonsense; next
qq 12b.9+ 67b.19+ c79.29 ;106 not used
qq 44b.9+ 67b.19+ c79.29 ;107 not used
qq 11b.9+ 67b.19+ c79.29 ;108 not used
qq 30b.9+ 67b.19+ c79.29 ;109 not used
qq 64b.9+ 67b.19+ c79.29 ;110 outcr; punch, test for proc
qq c6.9+ 74b.19+ 86b.29+ c11.39 ;111 lyn; clearR, lyn, float addr, stackVinRF
qq c35.9+ c17.19+132d.29+ c11.39 ;112 gier; address, total clear
[ hs, stack V in RF ]

```

```

[<then, ≥then, ≥, >]
d3: qq 36d.9+ c27.19 ; 0 a≠0∧b≠0, a in RF; srf, release
qq c27.9+ 36d.19+ 11b.29 ; 1 b in RF; release, srf, mt neg
qq c15.9+ c8.19+ 36d.29 [ 2 a and b in cell; exchange, top to
RF el, srf]
qq c27.9+ 11b.19+ c27.29 ; 3 a=0, b in RF; release, mt neg, release
qq c6.9+ 30d.19+ c27.29 ; 4 b in cell; clear R, srf, release
qq c27.9+ c27.19 ; 5 b= 0, a in RF; release, release
qq c27.9+ c8.19 ; 6 a in cell; release, top to RF rel
[≤then, >then, <, ≤]
qq 36d.9+ 11b.19+ c27.29 [ 7 a≠0∧b≠0, a in RF, srf, mt neg,
release]
qq c27.9+ 36d.19 ; 8 b in RF; release, srf
qq c8.9+ 36d.19 ; 9 a and b in cell; top to RF rel, srf
qq c27.9+ c27.19 ; 10 a=0, b in RF; release, release
qq c8.9+ c27.19 ; 11 b in cell; top to RF rel, release
qq c27.9+ 11b.19+ c27.29 ; 12 b=0, a in RF; release, mt neg, release
qq c27.9+ c6.19+ 30d.29 ; 13 a in cell; release, clear R, srf
[=then, ≠then]
qq 36d.9+ c27.19 ; 14 a≠0∧b≠0, a in RF; srf, release
qq c27.9+ 36d.19 ; 15 b in RF; release, srf
qq c8.9+ 36d.19 ; 16 a and b in cell; top to RF rel, srf
qq c27.9+ c27.19 ; 17 a=0, b in RF; release, release
qq c8.9+ c27.19 ; 18 b in cell; top to RF rel, release
qq c27.9+ c27.19 ; 19 b=0, a in RF; release, release
qq c27.9+ c8.19 ; 20 a in cell; release, top to RF rel
[=]
qq 36d.9+ c27.19 ; 21 a≠0∧b≠0, a in RF; srf, release
qq c27.9+ 36d.19 ; 22 b in RF; release, srf
qq c8.9+ 36d.19 ; 23 a and b in cell; top to RF rel, srf
qq c27.9+ c27.19 ; 24 a=0, b in RF; release, release
qq c6.9+ 38d.19+ c27.29+ c3.39 [ 25 b in cell; clear R, snn(f), release,
skip]
qq c27.9+ c27.19 ; 26 b=0, a in RF; release, release
qq c6.9+ c27.19+ 38d.29+ c3.39 [ 27 a in cell; clear R, release, snn(f),
skip]
[≠]
qq 36d.9+ 10b.19+ c27.29 [ 28 a≠0∧b≠0, a in RF; srf, mtneg LT,
release]
qq c27.9+ 36d.19+ 10b.29 ; 29 b in RF; release, srf, mt neg LT
qq c8.9+ 36d.19+ 10b.29 [ 30 a and b in cell; top to RF rel,
srf, mt neg LT]
qq 10b.9+ c27.19+ c27.29 [ 31 a=0, b in RF; mt neg LT, release,
release]
qq c6.9+ 39d.19+ c27.29 ; 32 b in cell; clear R, ann(f), release
qq 10b.9+ c27.19+ c27.29 [ 33 b=0, a in RF; mt neg LT, release,
release]
qq c6.9+ c27.19+ 39d.29 ; 34 a in cell; clear R, release, ann(f)

```

```

d40: qq 47d.9+ 24b.19 ;non-local goto: ps, goto nonlocal
d41: qq 42b.9+ 43b.19+ 46b.29+ 44b.39 ;assign; grf, gr, gm, grn(f)
d42: qq 50b.9+ 52b.29+ 51b.39 ;for assign; grf M, gm M, grn M
d43: qq c8.9+ 14b.19+ 59d.29 [last subscr; top to RF rel,
round, indexcall]
d44: qq 0b.9+ 71b.19+ c11.29+ 53d.39 [not first subscr; indexmult, stack
V in RF, ps()]
d45: qq c8.9+ 36d.19+ c27.29 [constant step; top to RF rel, srf,
release]
d46: qq c8.9+ 36d.19+ 32d.29 ;variable step; top to RF rel, srf, mkf
d47: qq c43.9+ c55.19+111b.29+102b.39 [simple element; while assign, get for label,
doabs, forlabel]
d50: qq 32d.9+ 32d.19+ c11.29 ;exp 2 and 3 ; mkf, mkf, stack V in RF
d51: qq c8.9+ c72.19+ 36d.29+ c11.39 [not constant low; top to RF rel,
no release, srf, stack V in RF]
d52: qq 45d.9+ 46d.19+ 45d.29+ c77.39 ;end constant b; pm, gm, pm, count p2
d53: qq 32d.9+ 31d.19+ 42d.29+ c67.39 [not first not last 2; mkf, arf, grf,
reset]
[1] qq 40d.9+ 32d.19+ 31d.29+ c70.39 ;not first last 2; arnf, mkf, arf, end decl
[2] qq 46d.9+ c67.19 ;first not last 2; gm, reset
[3] qq 40d.9+ c70.19 ;fir0t last 2; arnf, enddecl
[4] qq 42d.9+ 32d.19+ 42d.29+ 40d.39 ;not first not last 1; grf, mkf, grf, arnf
[5] qq 42d.9+ 32d.19+ 14b.29+ 43d.39 ;not first last 1; grf, mkf, round, gr
[6] qq c77.9+ 42d.19+ c77.29+ 45d.39 [first not last 1; count p2, grf,
count p2, pm]
[7] qq c77.9+ 14b.19+ 43d.29+ c77.39 [first last 1; count p2, round, gr,
count p2]
d54: qq 14b.9+ 43d.19+ c68.29 [finish bounds; round, gr, end const decl]
d55: qq125b.9+126d.19+127d.29+128d.39 [set array ident 1; take last used, ar_D,
gr_M, reserve space]
d56: qq124b.9+ c76.19 ;set array ident2; qq-1,hvs2, decl array
d57: qq 45b.9+ 46d.19 ;end const 2; pm, gm
d58: qq c16.9+ c15.19+ c8.29+ 103b.39 [real exp 1; clear UAUUV, exchange, top
to RF rel, areal]
d59: qq c14.9+ 40d.19+32d.29+ 122b.39 ;real exp2; stackVinUV, arnf, mkf, exp
d62: qq 62b.9+ c27.19+ 45d.29+ 63b.39 [int exp, exp in RF; grf c68, release,
pm, gm c54]
d63: qq 61b.9+ 45d.19+ 64b.29+ c27.39 [int exp, rad in RF; grf c54, pm,
gm c68, release]
d64: qq 45d.9+ 64b.19+ 45d.29+ 63b.39 [int exp in cell; pm, gm c 68, pm,
gm c54]
d66: qq 55d.9+ 21b.19+ 14b.29 [take value rounded; formal,
arnf(UA), round]
d68: qq 37d.9+ 56b.19+ 57b.29+ 58b.39 [int divide; dkf, srf c35 VNT, arf c35,
tkf-29, nkf 39]
d8: qq

d e18=c74, e16=k-e70+1 ;
e ;
d i=5e21, e22=e16-e15-1 ;
qq e15.9-e24.9+e18.19+e22.36+e22.34+205e13.39;
d e15=e16 ;
s ;

```

[19.1.66]

[PASS 6, page 1, GIER version]

b k=e15+e70, i=205e13, b30, c100, d40, a22;

d e17=i ; first location of control table

d d6=59 ; number of mode dependent input symbols

d d1=i+d6+d6-200 ; control table base, address for byte = 0

d d3=d1+279 ; base for operator table: d1 + max input byte - 1

d d2=49d3 ; base for operand table: d3 + no. of operators + 2

d i=26d2 ; first instruction: d2 + no. of operands

[The following are input byte values]

d d21=270 [CAR RET], d26=161 [no-type proc]

[The following are output byte values]

[GIER definitions, skipped for CDC]

d d10= 56 [integer], d11= 30 [round], d12= 36 [end RF exp] ;

d d13= 37 [end R exp], d14= 38 [end addr exp], d15= 37 [end string expr];

d d16= 42 [else adr exp], d17= 45 [end else adr exp], d18= 27 [end subscripts] ;

d d19= 29 [end switch des], d20= 34 [not], d22= 96 [end proc call] ;

d d23= 97 [end std. call], d24= 94 [prepare func], d25= 2 [begin call] ;

d d27= 4 [end proc], d28= 47 [prepare assign], d31= -2 [elseRF-elseadr] ;

d d32= -1 [else R-else adr]

s

[26. jan. 65]

[Pass 6, page 2, page 1 is on the tape for pass 7, to facilitate handling of CDC]

```
a20:can sd30 , hv c99 ; Initialize stack:
      grn s      M      ;   for s:= s - 1 while s ≠ d30 do
c60:ps s-1 , hv a20 ;   pack(store[s], 0, 41, 0);
      ;   go to in out 1;
c5: pmn(e1) Xt 1 ;   integer procedure input;
      hs e2      LA ;   R:= input:= next byte;
      hh s      ;
c65:pm (s)      DVX ;   procedure output(n); begin R:= n; output R end;
b6:
c66:pm[par 3] DX ;   procedure output par 3; begin R:= par 3; output R end;
c: hv e3      LQA ;   procedure output R; begin if relation is waiting then
      gr p3 , arn b13 ;   begin out(buffer); relation is waiting:= false end;
      hs e3      IQA ;   out(R);
      arn p3 , hv e3 ;   end output R;
b13:qq , ;
b7:
c9: qq 0 , pm [par 4]; stack operator of par 4: pos in input: neg in input:
      ;   left par input: not input: M:= OPERATOR TABLE[par 4];
c8: gm (b12) V 1 MA ;   stack operator: top operator:= topoperator + 1;
      ;   storewithmarks(M, ,a,, OPERATORSTACK[top operator]);
      ;   go to next symbol;
c7: qq (b12) t -1 ;   remove operator: right par input:
      ;   topoperator:= topoperator - 1;
c6: pmn(e1) XV 1 IZA ;   next symbol: inputsymbol:= input;
a1:
b2: pmn XV ;   output is normal:= true;
      hs e2      LA ;   if input symbol > ,min standard proc, then
      ga b1 V NT ;   begin input reference:= input symbol; input symbol:=
      ga b2 , hv a1 ;   part 1(STANDARDPROCEDURETABLE[input symbol])
b1: pm Xt d1 IRA ;   end; controlword:= CONTROLTABLE[input symbol];
      arn(b1) t -d6 LRC ;   markscontrolword:= marks CONTROLTABLE[input symbol];
      ga b4 , gt b5 ;   if parametermode ^ marks a(controlword) then
      ck 20 , ga b6 ;   begin controlword:= CONTROLTABLE[input -
      ;   ,number of mode dependent symbols,];
      gt b7 ;   marks controlword:= marks CONTROLTABLE[input -
      ;   ,number of mode dependent symbols,]
      hv c3 NB ;   end;
b5:
a3: arn p , hv[action]; par 1:= part 1(controlword); par 2:= part 2(controlword);
      ;   par 3:= part 3(controlword); par 4:= part 4(controlword);
      ;   if -, marksb(controlword) then go to repeat;
      ;   search finished: R:= OPERANDSTACK[top operand];
      ;   go to action [par 2];
c4: gm p ;   set top operand and repeat: OPERANDSTACK[topoperand]:= M;
c3:
b12:arn d5 , gt b9 ; repeat:
b4:
b9: bs [par 1]t[stackpriority]; stackpriority:= part 2 (OPERANDSTACK[topoperator]);
      hv a3 ;   if par 1 > stackpriority then go to search finished;
      ga b8 , tk 20 ;   p1:= part 1(OPERANDSTACK[topoperator]);
      ga b10 , tk 10 ;   p3:= part 3(OPERANDSTACK[topoperator]);
b8: hv [stackaction] ;   setpart 1(R) to:(part 4(OPERANDSTACK[top operator]));
      ;   go to stackaction [p1];
```

```

c10:ga b11 IZA ; standard proc 1 value after call:
; p4:= part 1(R); output is normal:= false;
c70:pm (b12) V -1 IQB ; after standard function integer argument:
; top operator:= top operator - 1;
; M:= OPERATORSTACK[top operator];
; marks M:= marks(OPERATORSTACK[top operator]);
; go to lab 0;
c11:pm (b2) IQB ; standard procedure in normal mode:
; M:= STANDARDPROCEDURETABLE[input reference];
; marks M:= marks STANDARDPROCEDURETABLE[input reference];
; lab 0: OUTPUT(part 4(M)); if marks b(M) then
;
;
; CDC
;
d i=i-e69 ;
qg ;
d i=i+e69-1 ;
a4: hv c75 LZA ; begin OUTPUT(part 3(M)); OUTPUT(part 2(M)); end;
; if output is normal then go to c75 else go to c14;
c12:pm p X ; stackaction 3:
; if -, comb 3(OPERANDSTACK[top operand]) then
; setpart 1(R) to: (,int,);
;
c13:hs c ; stackaction 1: OUTPUT(part 1(R));
c14:pm p X ; stackaction 2: R:= OPERANDSTACK[top operand];
; top operator:= top operator - 1;
; go to checkaction[p3];
b10:hv [checkaction] ;
c15:pm 1d2 , gm p ;
c16:ck 1 ; integer arithmetics from stack:
; if -, comb 1(R) then ALARM AND SET UNDECLARED;
; go to repeat;
;
c17:ck 2 , pm 2d2 ; real arithmetics from stack: M:= OPERANDTABLE[,real,];
; if comb 2(R) then go to set top operand and repeat;
; if -, comb 1(R) then ALARM AND SETUNDECLARED; go to repeat;
c18:ck 4 ; integer divide from stack:
; if -, comb 4(R) then ALARM AND SETUNDECLARED;
; go to repeat;
;
c19:ck 7 ; checkaction 4: prepare boolean assignment from stack:
; if -, comb 7(R) then ALARM;
; top operand:= top operand - 2; go to repeat;
c20:gm b13 IQA ; relation from stack:
; buffer:= M; M:= OPERANDTABLE[,boolean,];
; if comb 5(R) then OPERANDSTACK[top operand]:= M;
;
;
gm p LT ;
hs c2 NO ; if comb 6(R) = comb 5(R) then ALARM AND SETUNDECLARED;
;
hv c3 IQA ; relation is waiting:= true; go to repeat;
c21:ck 7 ; boolean-operator-from-stack:
; ulæseligt
;
; go to repeat;

```

```

c22:ck 8 ; undeclared operator from stack:
      hs c1 NO ; if -, comb 8(R) then ALARM;
      pm 13d2 , hv c4 ; M:= OPERANDTABLE[,undeclared,];
                        ; go to set top operand and repeat;
c23:ck 1 ; prepare real assign from stack:
a5:hs c1 NO ; if -, comb 1 (R) then ALARM;
c25:pp p-2 , hv c3 ; unstack repeat:
                        ; top operand:= top operand - 2; go to repeat;
c24:ck 3 ; prepare integer assign from stack:
      pm d11 DV LO ; if comb 3(R) then OUTPUT(,round,)
      ck -2 ; else if -, comb 1(R) then ALARM;
d i=i-e69-e69-e69 ; cdc: no round
      ck 1 ;
d i=i+e69-1 ;
      hs c1 NO ; top operand:= top operand - 2;
      hs c X LOA ; OUTPUT(,prepare assign,);
d i=i-e69 ; cdc: no round
      pp p-2 , ps c3-1 ; go to repeat;
      pm d28 D ;
      hv c X ;
c26:ga b11 IZA ; standard function integer argument:
      arn p , ck 3 ; p4:= part 1(R); output is normal:= false;
      pm d11 D ; if comb 3(OPERANDSTACK[top operand]) then
      hs c X LO ; OUTPUT(,round,); go to c70;
d i=i-e69-e69-e69 ; cdc: no round
      hv c70 ; comment after standard function integer argument;
c27:ck 1 V ; standard function arithmetic argument from stack:
                        ; if -, comb 1(R) then ALARM; go to lab 1;
c28:ck 29 ; standard function string argument from stack:
      hs c1 NO ; if -, comb 29(R) then ALARM;
b11:pm [p4] , gm p ; lab 1: OPERANDSTACK[top operand]:= OPERANDTABLE[p4];
      pa b3 , hv c6 ; skip bypasslabel:= true; go to next;
c96: ; bypasslabel:
b3:bs 1 , hv c40 ; if -,skip bypasslabel then goto output par 3 goto next
      pa b3 t 1 ; skip bypasslabel:= false;
      hv c6 ; go to next symbol;
c94:pa b4 t d2 ; procedure ALARM 2;
c92:hs e5 ; begin
      hr s1 , qq s+e41 ; PRINTTEKST(⟨proc.call or ident.⟩);
a17:pm s DX ; A:= s;
      qq , ud 9e4 ; select error by;
      hs e9 ; print(A);
      vy (8e4) , hr s1 ; end;
c30:ck 9 , pp p-2 ; designational expression: end goto:
      hs c1 NO ; top operand:= top operand - 2;
      ps c6-1 , hv c66 ; if -, comb 9(R) then ALARM;
                        ; OUTPUT PAR 3; go to next symbol;
c31:ck 10 ; procedure semicolon:
      hs c1 NO ; if -, comb 10(R) then ALARM;
c83:pp p-2 , ps c6-1 ; delete call: top operand:= top operand - 2;
      hv c66 , ; OUTPUT PAR 3; go to next symbol;
c32:tk 4 , ud b7 ; arith or relation in input: M:=
a6:pm (b6) NT ; OPERATORTABLE[if comb 3(R) then par 4 else par 3];

```



```

hv a7      LO      ; if comb 4(R) = comb 3(R) then
arn p      , ck 6   ; lab 3: begin if -, comb 6(OPERANDSTACK[top operand]
;           then ALARM;
hs c1      NO      ; M:= OPERANDTABLE[par 1]; comment undeclared;
pm (b4)    t d3     ; end;
a7: pp p-2  , hv c8  ; lab 4: toperand:= toperand - 2; go to stackoperator;
c33:ck 2    , hv a6  ; integer divide in input: M:= OPERANDTABLE[par 3];
;           go to if comb 2(R) then lab 4 else lab 3;
c34:ck 11   , hv a6  ; boolean operator in input: M:= OPERATORTABLE[par 3];
;           go to if comb 11(R) then lab 4 else lab 3;
c35:ck 12   ; left bracket in input:
hs c2      NO      ; if -, comb 12(R) then ALARM AND SET UNDECLARED;
hh c9      ; go to stack operator of par 4;
c93:ck 12   ; standard function array argument:
hs c1      NO      ; if -, comb 12(R) then ALARM;
hv b11     ; go to lab 1;
c36:pa b14  V d16   ; else expression:
;           inputsymbol:= ,else address expression,;
;           go to lab 5;
c37:pa b14  t d17   ; end else expression:
tk 20      , ck -6  ; input symbol:=,end else address expression,;
ga r2      , pp p-2 ; lab 5: top operand:= toperand - 2;
pmn p      X       ; current operand:= OPERANDSTACK[top operand];
hv         t a8     ; go to action 20 switch[R bits 20 to 22];
a8: hs c2   ; not allowed: ALARM AND SET UNDECLARED;
hv c6      ; go to next symbol;
[2a8]pm 13d2 , gm p  ; undeclared: OPERANDSTACK[top operand]:=,undeclared,;
hv c6      ; go to next symbol;
[4a8]ck 23  , ps 3   ; integer or integer procedure no parameters:
pm 1d2     , hv a9  ; if comb 23(current operand) then
; OPERANDSTACK[topoperand]:=,integer, else
; if -, comb 26(current operand) then
; ALARM AND SET UNDECLARED; go to RF case;
[6a8]ck 24  , ps 2   ; real or real procedure no parameters:
pm 2d2     , hv a9  ; if comb 24(current operand) then
; OPERANDSTACK[top operand]:=,real, else
; if 10, comb 26(current operand) then
; ALARM AND SET UNDECLARED; go to RFcase;
[8a8]ck 23  , ps 4   ; boolean or boolean procedure no parameters:
pm 3d2     , hv a10 ; if comb 23(current operand) then
; OPERANDSTACK[top operand]:= ,boolean, else
; if -, comb 27(current operand) then
; ALARM AND SET UNDECLARED; go to Rcase;
[10a8]ck23  , ps 5   ; label: if comb 23(current operand) then
pm 4d2     , hv a11 ; OPERANDSTACK[top operand]:= ,label, else
; if -, comb 28(current operand) then
; ALARM AND SET UNDECLARED;
; go to address case;

```

```

[12a8]ck23      , ps  6      ; string: if comb 23(current operand) then
      pm 5d2      , hv  a10   ;   OPERANDSTACK[top operand]:= ,string,;
                                   ;   if -, comb 29(current operand) then
                                   ;   ALARM AND SET UNDECLARED; go to Rcase;

[14a8]arnp      , ck  25     ; nonsense:
      ps  1      , pm  6d2   ;   if comb 25(current operand) then
                                   ;   OPERANDSTACK[top operand]:= ,nonsense, else
                                   ;   if -, comb 26(current operand) then
                                   ;   ALARM AND SET UNDECLARED;

a9: qq (b14)    V d31        ; RF case: input symbol:= RFversion(input symbol);
                                   ;   go to address case;

a10:qq (b14)    t d32        ; Rcase: input symbol:= R version(input symbol);
a11:gm  p      V      LO     ; address case:
      ck  s      ;   OUTPUT(input symbol);
      hs  c2      NO        ;   go to next symbol;

b14:pm[input symbol] DX      ;
      ps  c6-1    , hv  c      ;

c38:pp  p-2      , pm  p      ; colon equal input:
      tl  16      ITA        ;   top operand:= top operand - 2;
      tl  2      V      LO    ;   M:= OPERANDSTACK[top operand];
      ck  2      V      NTA    ;   if (comb 15(R) ^ comb16(R) ^ -, comb 17(M)) v
      tl  1      LTA        ;   (comb 15(R) ^ -, comb 16(R) ^ -, comb 19(M)) v
      ck  -1      ;   (-, comb 15(R) ^ comb 16(R) ^ -, comb 18(M)) v
      hs  c2      NO        ;   -, comb 17(R) then ALARM AND SET UNDECLARED;
      ps  c9-1    , hv  c66   ;

                                   ;   OUTPUT PAR 3; go to stack operator of par 4;

c95:tk  2      , tk  -3      ; stepelement: stepelement and do:
c39:tk  2      , pp  p-2     ; forelements input:
      hs  c1      NT        ;   topoperand:= top operand - 2;
      hv  c40      NO        ;   if -, comb 1(R) then ALARM;
      arn p      , ck  2     ;   if comb 1(R)  $\equiv$  comb 2(R) then go to c40;
d i=i-e69      ;   cdc: no round
a13:pm  d11    D      LO     ;   if comb 2(OPERANDSTACK[topoperand]) then
      hs  c      X      LO    ;   OUTPUT(,round,);
d i=i-e69-e69  ;   cdc: no round
c40:ps  c6-1    , hv  c66   ; output par 3 go to next: expression in parameter mode:
                                   ; it statement: end elset: for: OUTPUT PAR 3;
                                   ;   go to next symbol;

c41:tk  2      , pp  p-2     ; right bracket input:
      hs  c1      NT        ;   top operand:= top operand - 2;
      pm  d11    D      LO    ;   if -, comb 1(R) then ALARM;
      hs  c      X      LO    ;   if -, comb 1(R)  $\equiv$  comb 2(R)) then OUTPUT(,round,);
d i=i-e69-e69  ;   cdc: no round
      pm  p      X          ;   current operand:= OPERANDSTACK[topoperand];
      tk  31      IOB       ;   operand array:= -, (comb 30(current operand)  $\equiv$ 
                                   ;   comb 31(current operand));
      hv  a12      NT        ;   if comb 30(current operand) then
      pm (p-1)    DX  -1     ;   begin
      hs  c91      NZ        ;   set part 1(OPERANDSTACK[top operator - 1])
                                   ;   to:(OPERANDSTACK[topoperator-1]-1);
      pm  d19      DVX  NOB   ;   if part 1(OPERANDSTACK[top operator - 1]  $\neq$  0
                                   ;   then ALARM 1
                                   ;   end;

```

```

a12:pm d18 DX ;
hs c ; OUTPUT(
arn p , tk 33 ; if operandarray then, array, else, switch,);
ck -7 , ga r1 ; OPERANDSTACK[top operand]:=
pm t d2 ; OPERANDTABLE[bit 33 to 35(current operand)];
gm p , hv c7 ; go to remove operator;
c42:tk 15 , ck -8 ; first assign input:
ga r2 , tk -29 ; OPERANDSTACK[top operand]:=
hs c1 LZ ; OPERANDTABLE[bit 15 to 16(R)];
pm t d2 ; if -, (comb 15(R) v comb 16(R) v comb 17(R)) then
gm p , ps c9-1 ; ALARM; OUTPUT PAR 3;
hv c66 ; go to stackoperator of par 4;
c43:qq d20 , hs c65 ; implication input: OUTPUT(, ,); par 1:= 5;
pa b4 t 5 ; R:= OPERANDSTACK[top operand];
arn p , hv c34 ; go to c34; comment boolean operator in input;
c44:qq (b13) V -6 NQA ; then input: if relation is waiting then
; begin comment change to then version;
; buffer:= buffer - 6 end else
c45:hs c66 ; while element input:
pm p X ; OUTPUT PAR 3;
ck 7 ;
hs c1 NO ; if -, comb 7(OPERANDSTACK[top operand]) then ALARM;
c85:pp p-2 , hv c6 ; remove operand next symbol: do:
; topoperand:= topoperand - 2;
; go to next symbol;
c46:ck 32 ; colonequal for:
hs c2 NO ; if -, comb 32(R) then ALARM AND SET UNDECLARED;
ps c6-1 , hv c66 ; OUTPUT PAR 3; go to next symbol;
c47:qq (p-3) t -1 ; subscript comma: set part 1(OPERANDTABLE[topoperand-3])
; to: (part 1(OPERANDSTACK[topoperand]-3))-1);
c48:tk 2 ; end bounds: bound comma: if -, comb 1(R) then ALARM;
hs c1 NT ; top operand:= top operand - 2;
pp p-2 , hv a13 ; if -, (comb 1(R) ≡ comb 2(R)) then OUTPUT(,round,);
; OUTPUT PAR 3; go to next symbol;
c49:tk 2 , pp p-2 ; boundcolon:
hs c1 NT ; top operand:= top operand - 2;
pm d11 D LO ; if -, comb 1(R) then ALARM;
hs c X LO ; if -, (comb 1(R) ≡ comb 2(R)) then OUTPUT(,round,);
d i=i-e69-e69 ; cdc: no round
hv c6 ; go to next symbol;
c50:hs c66 ; output par 3 copy 1:
hs c5 , hs c ; OUTPUT PAR 3;
hv c6 ; OUTPUT(input);
; go to next symbol;
c51:hs c66 ; endpass:
pm d4 X -1 ; OUTPUT PAR 3;
hv r-1 LZ ; information 1:= ,max height of operatorstack,;
nt d5 , it (r-2) ; information 2:= ,max height of operandstack,;
pa 2e4 ; if no running then go to initialize translator;
pm e25-e71+40 X -1 ; go to pass 7;
hv r-1 LZ ;
nt d4 , it (r-2) ;
pa 3e4 , hh e94 ;

```

[Pass 6, page 8]

```

c52:hs c5      , ga b15 ; begcall:
      hs c5      , ga b16 ;   number of actuals:= input
      pmf r      , IRB   ;   procedurekind:= input;
      hv a14      , NT    ;   parametermode:= true;
      arn(b16)    , ga r1  ;   if procedurekind ,min standard proc, then
      arn         t d1    ;   R:= CONTROLTABLE[part 1(STANDARDPROCEDURETABLE
                        ;   [procedurekind])] else
      hv a15      ;   begin
a14:hs c5      , ga b17 ;   number of formals:= input;
      hs c5      , ga b18 ;   relative address:= input;
      hs c5      , ga b19 ;   blocknumber:= input;
      it d22      ;   callbyte:= ,end proc call,;
      pa b20      ;
b16:pmn[proc.kind] Xt d1 ;
      d i=i-e69-e69-e69 ;
      is (b16)    , it s+d22 ; CDC
      d i=i+e69+e69+e69-1 ; R:= CONTROLTABLE[procedurekind]
                        ; end;
a15:ga b4      , ck 20   ; par 1:= part 1(R);
      gt r, hv[callaction] ; go to callaction[part 4(R)];
                        ; comment c53, c54, c67, c55, c56, c87, c57, or c59;
c53:qq d24      , hs c65 ; standard procedure 1 value in call: OUTPUT(,prep func,);
      bs (b15)    t 1     ; if number of actuals 1 then go to c59;
                        ; comment procedure call alarm;
      hv c59      ; top operator:= top operator + 1;
      pm (b16)     IQB    ; storewithmarks(STANDARDPROCEDURETABLE[procedurekind],
      gm (b12)    t 1 MQB ; marks STANDARDPROCEDURETABLE[procedurekind],
      pm (b4)     IRB    ; OPERATORSTACK[topoperator]);
      hv c8        ; M:= OPERATORSTACK[par1]; parametermode:= false;
                        ; go to stackoperator;
c54:qq (b4)      V 3     ; procedure call or assign in call:
                        ; par 1:= par 1 + 3; go to c55;
c67:qq (b4)      t 22    ; no type in call only in call: par 1:= par 1 + 22;
c55:arn(b15)     D       ; not formal call only in call:
b17:nc          , hv c59 ; if number of actuals  $\neq$  number of formals then
                        ; go to c59;
c56:qq (b4)      t -22   ; formal procedure in call:
                        ; par 1:= par1 - 22;
a16:
c87:qq d25      , hs c65 ; formal procedure no type in call:
b15:srn         Dt 1     ; OUTPUT(.beg call,);
      hs c       t       ; OUTPUT(number of actuals + 1);
      hv c75      ; go to c75; comment stack operand from par 1;
c57:arn(b16)    , ck 10   ; standard procedure several parameters in call:
      ga b19      , ck 10 ; R:= STANDARDPROCEDURETABLE[procedurekind];
      pa b20      t d23   ; blocknumber:= part 2(R);
                        ; callbyte:= ,end standard call,;
      ga b18      , hv a16 ; relative address:= part 3(R); go to c87;
c58:hs c66      ; standard procedure in parametermode:
      pmn(b2)     , cl -10 ; OUTPUT PAR 3;
      cln -10     , hs c   ; M:= STANDARDPROCEDURETABLE[input reference];
      cln -10     , hs c   ; OUTPUT(part 3(M));
      d i=i-e69    ; OUTPUT(part 2(M)); comment Not for CDC;
      hv c6        ; go to next symbol;
c59:hs c92      ; c59: procedure call alarm:
      pm d2       , hh c75 ; ALARM 2;
                        ; M:= OPERANDTABLE[,undeclared,];
                        ; go to c75h; comment go to next symbol;

```



```

c76:hs  c66                ; bounds: OUTPUT PAR 3;
      hs  c5      , hs  c  ;   OUTPUT(input);
      hs  c5      , hs  c  ;   OUTPUT(input);
      hv  c6                ;   go to next symbol;
c77:hs  c5      ,          ;   call or assign normal mode;
      qq (b4)  t -3 NZ      ;   if input  $\neq$  0 then par 1:= par 1 - 3;
      hh  c73                ;   go to c73h;
c78:hs  c66                ; constant normal mode: OUTPUT PAR 3;
      hs  c5      , hs  c  ;   For CDC:
d i=i+e69-1          ;   OUTPUT(input);
      hs  c5      , hs  c  ;   OUTPUT(input);
      hs  c5      , hs  c  ;   OUTPUT(input);
      hv  c74                ;   go to c74;
c79:hs  c5      ,          ; call only normal mode:
      hs  c94      NZ      ;   if input  $\neq$  0 then ALARM 2a;
      hh  c73                ;   go to c73h;
c80:hs  c5      , hs  c66   ; normal identifiers parameter mode:
                        ;   R:= input;
                        ;   OUTPUT PAR 3;

c84:hs  c5      , hs  c  ; copy 2 next symbol: OUTPUT(input);
c99:hs  c5      , hs  c  ; copy next symbol: OUTPUT(input);
      hv  c6                ;   go to next symbol;
c81:hs  c66                ; constants parameter mode: OUTPUT PAR 3;
      hs  c5      , hs  c  ;   For CDC
d i=i+e69-1          ;   OUTPUT(input);
      hs  c5      , hs  c  ;   OUTPUT(input);
      hs  c5      , hs  c  ;   OUTPUT(input);
      hv  c84                ;   go to c84;
c82:qq (e4)  t      1      ; CR: CRcounter:= CRcounter + 1;
      hv  c6                ;   go to next symbol;
d i=i-e69          ;   For CDC:
      ps  c6-1    , hv  c66 ;   OUTPUT(CRcounter);
d i=i+e69-1          ;   go to next symbol;

```

```

c2: pm d2      , gm p      ; procedure ALARM AND SET UNDECLARED;
                                ; begin OPERANDSTACK[topoperand]:= undeclared;
                                ; ALARM end;
c1: hs e5      ; procedure ALARM;
    hv a17     , qq s+e42 ; begin PRINTTEXT({<type>}); PRINT(s) end;
c91:hs e5      ; procedure ALARM 1;
    hv a17     , qq s+e40 ; begin PRINTTEXT({<subscript>}); PRINT(s) end;
                                ; actual parameter expression from stack:
c29:ck -4      , tk -7     ; R:= bits 36 to 38 (R);
    ga b25     , pp p-2    ; top operand:= top operand - 2;
    ca 0       , hs c1     ; if R = 0 then ALARM;
b25:pmn[end expr]Xta21    ; OUTPUT(END_EXPR[R]);
    ps c6-1    , hv c      ; goto next symbol;
d a21=i-1      ; END_EXPR
    qq d12     ; 1 end RF expression
    qq d13     ; 2 end R expression
    qq d14     ; 3 end address expression
    qq d15     ; 4 end string expression
c100:ck 6      , hv c27    ; st. func. boolean argument:
d5: qq c13 +1.19+777.39 ; operatorstack pos 0

```

[The following instructions are overwritten by the stack]

```

                                ; start pass 6:
c86:                                ; tables from drum:
    gs b21     , ud 5e4      ; s:= save no:= number of std idents;
a18:vk p1      , pp p1      ; p:= std table track; core:= top table place;
    lk 40e25 t -40          ; rep: p:= p - 1; core:= core - 40;
    ps s-40    , vk p1      ; to core(p, core); s:= s - 40;
    bs s       , hh a18     ; if s > 0 then go to rep;
    hs c5      , ga b22     ; R:= stackref 0:= input;
b21:pp -1      , hs e3      ; p:= save no; output(R);
b22:srn -1     D           ; stackref 0:= -stackref 0x2(-20);
    ck 20      , gr b22     ; R:=0;
    arn 2e4    , mb b24     ; pack(R, 10, 19, first std proc track);
a19:                                ; modify table:
b23:pm 40e25 t -1          ;
    pm b22     VX LA       ; for p:= p, p-1 while p > 0 do
    ac (b23) V           ; table[p]:= table[p] +
    ac (b23) X MB        ; (if bit(40, table[p]) = 0 then R
    pp p-1     , bs p      ; else stackref 0 + bit 41 - bit 40);
    hv a19     ;
    ps (b23)   , pp d4     ; s:= stack top + 1;
    pi 1.6     , hv c60    ; p:= address(operandstack[0]);
                                ; normal:= t; condition waiting:= f;
                                ; go to initialize stack;
b24:qq 0       t 511      ; mask for first std proc track;
d d30=-d5      ; -address(operator stack[0]);
d29:

```

s

[9.12.65]

[PASS 6, page 12, CONTROL TABLE 1, PARAMETERMODE CDC]

[Action Output]

```
d i=e17 ;
qq      c58.19+13.29 f ; string in real out
qq      c58.19+13.29 f ; no par. bool out
qq      c58.19+13.29 f ; more par. int.out
qq      c58.19+13.29 f ; more par. real out
qq      c58.19+13.29 f ; more par. bool out
qq      c58.19+142.29f ; standard integer
qq      c58.19+13.29 f ; array, int. out
qq      c58.19+13.29 f ; no par. int. out
qq      c58.19+13.29 f ; int. in, no type
qq      c81.19+ 0.29 f ; lit string
qq      c81.19+ 1.29 f ; lit.int.
qq      c81.19+ 2.29 f ; lit.real
qq      c81.19+ 3.29 f ; lit. Boolean
qq      c80.19+ 4.29 f ; own int.
qq      c80.19+ 5.29 f ; own real
qq      c80.19+ 6.29 f ; own Boolean
qq      c80.19+ 7.29 f ; switch
qq      c80.19+ 7.29 f ; array int
qq      c80.19+ 7.29 f ; array real
qq      c80.19+ 7.29 f ; array Bool
qq      c80.19+ 7.29 f ; proc. no type
qq      c80.19+ 7.29 f ; proc. no int.
qq      c80.19+ 7.29 f ; proc. no real
qq      c80.19+ 7.29 f ; proc. no Bool
qq      c80.19+ 8.29 f ; simple int
qq      c80.19+ 9.29 f ; simple real
qq      c80.19+10.29 f ; simple Bool
qq      c80.19+11.29 f ; label
qq      c80.19+ 7.29 f ; formal array int.
qq      c80.19+ 7.29 f ; formal array real
qq      c80.19+ 7.29 f ; formal array Bool
qq      c80.19+ 7.29 f ; formal proc. no type
qq      c80.19+ 7.29 f ; formal proc. no int.
qq      c80.19+ 7.29 f ; formal proc. real
qq      c80.19+ 7.29 f ; formal proc. Bool
qq      c80.19+ 7.29 f ; formal simple int.
qq      c80.19+ 7.29 f ; formal simple real
qq      c80.19+ 7.29 f ; formal simple Bool
qq      c80.19+ 7.29 f ; formal label
qq      c80.19+ 7.29 f ; formal switch
qq      c80.19+ 7.29 f ; formal string
qq      c80.19+ 7.29 f ; proc. call or assign int
qq      c80.19+ 7.29 f ; proc.call or assign real
qq      c80.19+ 7.29 f ; proc.call or assign Bool
qq      c40.19+12.29 f ; expression
qq      c62.19      f ; end call
qq      c63.19      f ; beg exp
qq      c80.19      f ; undeclared
qq      c58.19+13.29 f ; l s string
qq      c58.19+13.29 f ; l st
qq      c58.19+13.29 f ; output no param.
qq      c58.19+13.29 f ; several par. no type
qq      c58.19+13.29 f ; several par. nonsense
qq      c58.19+13.29 f ; arith in. real out
qq      c58.19+13.29 f ; string in, nons. out
qq      c58.19+13.29 f ; string in, Bool. out
qq      c58.19+13.29 f ; arith in, int. out
qq      c58.19+13.29 f ; int.in, nons. out
qq      c58.19+13.29 f ; arith. in, real out
```


[PASS 6, page 13, CONTROL TABLE 2, NORMAL MODE CDC]

[Type Action Output Call with par.]

```
[141] qq 48d3+c59.19      +c53.39 f, ; bool in. real out
[142] qq 25d2+c11.19      +c59.39 f, ; no par. bool. out
[143] qq 1d2+c59.19       +c57.39 f, ; more par. int.out
[144] qq 2d2+c59.19       +c57.39 f, ; more par. real out
[145] qq 3d2+c59.19       +c57.39 f, ; more par. bool.out
[146] qq 1d2+c11.19       +c59.39 f, ; standard integer
[147] qq 46d3+c59.19      +c53.39 f, ; array, int. out
[148] qq 23d2+c11.19      +c59.39 f, ; no par. int. out
[149] qq 47d3+c59.19      +c53.39 f, ; int. in, no type
[150] qq 5d2+c78.19+14.29      f, ; lit.string
[151] qq 1d2+c78.19+15.29      f, ; lit.int.
[152] qq 2d2+c78.19+16.29      f, ; lit.real
[153] qq 3d2+c78.19+17.29      f, ; lit.Boolean
[154] qq 1d2+c73.19+18.29+c59.39 f, ; own integer
[155] qq 2d2+c73.19+19.29+c59.39 f, ; own real
[156] qq 3d2+c73.19+20.29+c59.39 f, ; own Boolean
[157] qq 11d2+c72.19+21.29+c59.39 f, ; switch
[158] qq 8d2+c71.19+22.29+c59.39 f, ; array integer
[159] qq 9d2+c71.19+23.29+c59.39 f, ;      real
[160] qq 10d2+c71.19+24.29+c59.39 f, ;      Boolean
[161] qq 7d2+c79.19+27.29+c67.39 f, ; proc. no type
[162] qq 23d2+c79.19+25.29+c55.39 f, ;      integer
[163] qq 24d2+c79.19+26.29+c55.39 f, ;      real
[164] qq 25d2+c79.19+27.29+c55.39 f, ;      Boolean
[165] qq 1d2+c73.19+28.29+c59.39 f, ; simple integer
[166] qq 2d2+c73.19+29.29+c59.39 f, ;      real
[167] qq 3d2+c73.19+30.29+c59.39 f, ;      Boolean
[168] qq 4d2+c73.19+31.29+c59.39 f, ; label
[169] qq 14d2+c73.19+32.29+c59.39 f, ; formal array integer
[170] qq 15d2+c73.19+33.29+c59.39 f, ;      real
[171] qq 16d2+c73.19+34.29+c59.39 f, ;      Boolean
[172] qq 7d2+c73.19+37.29+c87.39 f, ; formal proc.no type
[173] qq 23d2+c73.19+35.29+c56.39 f, ;      integer
[174] qq 24d2+c73.19+36.29+c56.39 f, ;      real
[175] qq 25d2+c73.19+37.29+c56.39 f, ;      Boolean
[176] qq 1d2+c73.19+38.29+c59.39 f, ; formal simple integer
[177] qq 2d2+c73.19+39.29+c59.39 f, ;      real
[178] qq 3d2+c73.19+40.29+c59.39 f, ;      Boolean
[179] qq 4d2+c73.19+41.29+c59.39 f, ; formal label
[180] qq 11d2+c72.19+21.29+c59.39 f, ; formal switch
[181] qq 5d2+c73.19+61.29+c59.39 f, ; formal string
[182] qq 20d2+c77.19+25.29+c54.39 f, ; proc.call or ass.int.
[183] qq 21d2+c77.19+26.29+c54.39 f, ;      real
[184] qq 22d2+c77.19+27.29+c54.39 f, ;      Bool.
[185] qq      c69.19          f, ; expression
[186] qq      2                , ; end call
[187] qq 39d3+c64.19+42.29      f, ; beg exp
[188] qq 13d2+c73.19          +c87.39 f, ; undeclared
[189] qq 7d2+c11.19          +c59.39 f, ; l sstreng
[190] qq 24d2+c11.19          +c59.39 f, ; l st
[191] qq 6d2+c11.19          +c59.39 f, ; output no param.
[192] qq 7d2+c59.19          +c57.39 f, ; several par. no type
[193] qq 6d2+c59.19          +c57.39 f, ; several par. nonsense
[194] qq 40d3+c59.19          +c53.39 f, ; arith.in, real out
[195] qq 41d3+c59.19          +c53.39 f, ; string in, nons. out
[196] qq 42d3+c59.19          +c53.39 f, ; string in, Bool. out
[197] qq 43d3+c59.19          +c53.39 f, ; arith.in, int. out
[198] qq 44d3+c59.19          +c53.39 f, ; int.in, nonsense out
[199] qq 45d3+c59.19          +c53.39 f, ; arith.in, real out
```

[mode independent symbols originating in pass 3, values rel. to d50 in 3]
[Priority Action Output or operator]

[2.12.65]

[PASS 6, page 15, CONTROL TABLE 4 CDC]

[MODE INDEPENDENT SYMBOLS FROM PASS 4, VALUES RELATIVE TO d22]

[PRIORITY ACTION OUTPUT]

```
[ 0] qq      c89.19+ 57.29      f ; end local
[ 1] qq      c96.19+ 58.29      f ; bypasslabel
[ 2] qq      c40.19+ 59.29      f ; goto bypasslabel
[ 3] qq      c40.19+ 60.29      f ; labelcolon
[ 4] qq 2.9+c49.19              ; boundcolon
[ 5] qq 2.9+c48.19+125.29        ; end bounds integer
[ 6] qq 2.9+c48.19+126.29        ; end bounds real
[ 7] qq 2.9+c48.19+127.29        ; end bounds Boolean
[ 8] qq 3.9+c44.19+ 47.29        ; thenst = then ex
[ 9] qq      c40.19+ 62.29      f ; else st
[10] qq      c46.19+ 63.29      f ; := for
[11] qq 3.9+c95.19+ 64.29        ; stepelem
[12] qq      c42.19+ 65.29+38d3.39 f ; first:=
[13] qq 3.9+c39.19+ 66.29        ; while
[14] qq 3.9+c45.19+ 67.29        ; while element
[15] qq      c68.19              f ; prepare assign
[16] qq 3.9+c39.19+ 68.29        ; simple for
[17] qq 2.9+ c6.19              ; end assign
[18] qq      c85.19              f ; do
[19] qq      c40.19+ 69.29      f ; while label
[20] qq      c38.19+ 65.29+38d3.39 f ; :=
[21] qq 2.9                      ; end ex
[22] qq 2.9+c30.19+109.29        ; end des
```

[MODE INDEPENDENT SYMBOLS FROM PASS 5. VALUES RELATIVE TO d10]

```
[ 0] qq      c50.19+ 70.29      f ; local switch
[ 1] qq      c50.19+ 71.29      f ; local proc
[ 2] qq      c40.19+ 72.29      f ; local label
[ 3] qq      c40.19+ 73.29      f ; blind
[ 4] qq      c51.19+ 74.29      f ; end pass
[ 5] qq      c82.19+130.29      f ; CR
[ 6] qq      c50.19+ 75.29      f ; beg block
[ 7] qq      c50.19+ 76.29      f ; beg proc
[ 8] qq      c61.19              f ; end proc
[ 9] qq      c50.19+ 77.29      f ; take value integer
[10] qq      c50.19+ 78.29      f ; take value real
[11] qq      c50.19+ 79.29      f ; take value Boolean
[12] qq      c52.19+ 80.29      f ; beg call
[13] qq      c40.19+ 81.29      f ; end block
[14] qq      c76.19+ 82.29      f ; bounds
[15] qq      c88.19+138.29      f ; beg local
```

[19.1.66]

[PASS 6, page 16. OPERATOR TABLE CDC]

[i=2d3

ACTION PRIORITY CHECK OUTPUT OUTPUTSYMBOL

```
[ 2] qq c14+ 2.19+c25.29      ; prep. undecl. ass.
[ 3] qq c14+ 3.19+c22.29      ; undecl. 3
[ 4] qq c14+ 4.19+c22.29      ; undecl. 4
[ 5] qq c14+ 5.19+c22.29      ; undecl. 5
[ 6] qq c14+ 6.19+c22.29      ; undecl. 6
[ 7] qq c14+ 7.19+c22.29      ; undecl. 7
[ 8] qq c14+ 8.19+c22.29      ; undecl. 8
[ 9] qq c14+ 9.19+c22.29      ; undecl. 9
[10] qq c14+10.19+c22.29      ; undecl. 10
[11] qq c14+11.19+c22.29      ; undecl. 11
[12] qq c13+ 9.19+c16.29+ 83.39 ; i+ : +
[13] qq c13+ 9.19+c16.29+ 84.39 ; i- : -
[14] qq c13+10.19+c16.29+ 85.39 ; ix : x
[15] qq c13+ 9.19+c16.29+105.39 ; neg: neg
[16] qq c14+ 9.19+c16.29      ; pos
[17] qq c13+ 9.19+c17.29+ 83.39 ; r+ : +
[18] qq c13+ 9.19+c17.29+ 84.39 ; r- : -
[19] qq c13+10.19+c17.29+ 85.39 ; rx : x
[20] qq c13+10.19+c17.29+ 86.39 ; /: /
[21] qq c12+11.19+c17.29+ 89.39 ; r↑ : ↑ real
[22] qq c13+10.19+c18.29+ 87.39 ; ⋮ : ⋮
[23] qq c12+11.19+c16.29+ 89.39 ; i↑ : ↑real
[24] qq c14+ 8.19+c20.29+ 96.39 ; < : <
[25] qq c14+ 8.19+c20.29+ 97.39 ; ≤ : ≤
[26] qq c14+ 8.19+c20.29+ 98.39 ; = : =
[27] qq c14+ 8.19+c20.29+ 99.39 ; ≥ : ≥
[28] qq c14+ 8.19+c20.29+100.39 ; > : >
[29] qq c14+ 8.19+c20.29+101.39 ; ≠ : ≠
[30] qq c13+ 6.19+c21.29+102.39 ; ^ : ^
[31] qq c13+ 5.19+c21.29+103.39 ; v : v
[32] qq c13+ 3.19+c21.29+104.39 ; ≡ : ≡
[33] qq c13+ 7.19+c21.29+d20.39 ; -, : -,
[34] qq c14+ 2.19+c24.29+d28.39 ; prep. int. ass. : prepare assign
[35] qq c13+ 2.19+c23.29+d28.39 ; prep.real ass. : prepare assign
[36] qq c13+ 2.19+c19.29+d28.39 ; prep. Bool. ass. : prepare assign
[37] qq      1.19              ; (
[38] qq c13+ 2.19+ c3.29+107.39 ; := : :=
[39] qq c14+ 2.19+c29.29[RESULT TYPE] ; actual parameter
[40] qq c10+ 2.19+c27.29+2d2.39 ; arith.in, val. out: real
[41] qq c10+ 2.19+c28.29+6d2.39 ; string in.nons. out: nonsense
[42] qq c10+ 2.19+c28.29+3d2.39 ; string.Bool. out: Boolean
[43] qq c10+ 2.19+c27.29+1d2.39 ; arith.in, int. out: integer
[44] qq c26+ 2.19+c27.29+6d2.39 ; int.in, nons. out: nonsense
[45] qq c10+ 2.19+c27.29+2d2.39 ; arith.in, val. out: real
[46] qq c10+ 2.19+c93.29+1d2.39 ; array in, int. out: integer
[47] qq c26+ 2.19+c27.29+7d2.39 ; int. in, no type: no type
[48] qq c10+ 2.19+c28.29+2d2.39 ; string in, real out: real
```

d8:

[TABLE HAS 47 WORDS, FROM 2d3 TO 48d3]

```
d a2=e69+e69, a2=a2+a2+a2+a2      ; e69= if CDC then 1 else 0
d a2=a2+a2+a2+a2+a2+a2+a2+a2      ; a2=64xe69
b k=e15+e70+a2, i=e17              ; begin drum block
```

[25. Jan. 65]

[PASS 6, page 17, CONTROL TABLE 1, PARAMETERMODE GIER]

[Action Output]

```
qq      c58.19+89.29 f ; bool in real out
qq      c58.19+89.29 f ; no par. bool out
qq      c58.19+89.29 f ; more par. int.out
qq      c58.19+89.29 f ; more par. real out
qq      c58.19+89.29 f ; more par. bool out
qq      c58.19+86.29 f ; standard integer
qq      c58.19+89.29 f ; array, int. out
qq      c58.19+89.29 f ; no par. int. out
qq      c58.19+89.29 f ; int. in, no type
qq      c81.19+90.29 f ; lit string
qq      c81.19+90.29 f ; lit.int.
qq      c81.19+90.29 f ; lit.real
qq      c81.19+90.29 f ; lit. Boolean
qq      c80.19+87.29 f ; own int.
qq      c80.19+87.29 f ; own real
qq      c80.19+87.29 f ; own Boolean
qq      c80.19+88.29 f ; switch
qq      c80.19+88.29 f ; array int
qq      c80.19+88.29 f ; array real
qq      c80.19+88.29 f ; array Bool
qq      c80.19+88.29 f ; proc. no type
qq      c80.19+88.29 f ; proc. no int.
qq      c80.19+88.29 f ; proc. no real
qq      c80.19+88.29 f ; proc. no Bool
qq      c80.19+86.29 f ; simple int
qq      c80.19+86.29 f ; simple real
qq      c80.19+86.29 f ; simple Bool
qq      c80.19+86.29 f ; label
qq      c80.19+88.29 f ; formal array int.
qq      c80.19+88.29 f ; formal array real
qq      c80.19+88.29 f ; formal array Bool
qq      c80.19+88.29 f ; formal proc. no type
qq      c80.19+88.29 f ; formal proc. no int.
qq      c80.19+88.29 f ; formal proc. real
qq      c80.19+88.29 f ; formal proc. Bool
qq      c80.19+88.29 f ; formal simple int.
qq      c80.19+88.29 f ; formal simple real
qq      c80.19+88.29 f ; formal simple Bool
qq      c80.19+88.29 f ; formal label
qq      c80.19+88.29 f ; formal switch
qq      c80.19+88.29 f ; formal string
qq      c80.19+88.29 f ; proc. call or assign int
qq      c80.19+88.29 f ; proc.call or assign real
qq      c80.19+88.29 f ; proc.call or assign Bool
qq      c40.19+85.29 f ; expression
qq      c62.19      f ; end call
qq      c63.19      f ; beg exp
qq      c80.19      f ; undeclared
qq      c58.19+89.29 f ; l s string
qq      c58.19+89.29 f ; l st
qq      c58.19+89.29 f ; output no param.
qq      c58.19+89.29 f ; several par. no type
qq      c58.19+89.29 f ; several par. nonsense
qq      c58.19+89.29 f ; arith in, real out
qq      c58.19+89.29 f ; string in, nons. out
qq      c58.19+89.29 f ; string in, Bool. out
qq      c58.19+89.29 f ; arith in, int. out
qq      c58.19+89.29 f ; int.in, nons. out
qq      c58.19+89.29 f ; arith. in, real out
```

[PASS 6. page 18, CONTROL TABLE 2, NORMAL MODE GIER]

[Type Action Output Call with par.]

```
[141] qq 48d3+c59.19      +c53.39 f, ; bool in. real out
[142] qq 25d2+c11.19      +c59.39 f, ; no par. bool. out
[143] qq 1d2+c59.19       +c57.39 f, ; more par. int.out
[144] qq 2d2+c59.19       +c57.39 f, ; more par. real out
[145] qq 3d2+c59.19       +c57.39 f, ; more par. bool.out
[146] qq 1d2+c11.19       +c59.39 f, ; standard integer
[147] qq 46d3+c59.19      +c53.39 f, ; array, int. out
[148] qq 23d2+c11.19      +c59.39 f, ; no par. int. out
[149] qq 47d3+c59.19      +c53.39 f, ; int. in, no type
[150] qq 5d2+c78.19+95.29      f, ; lit.string
[151] qq 1d2+c78.19+95.29      f, ; lit.int.
[152] qq 2d2+c78.19+95.29      f, ; lit.real
[153] qq 3d2+c78.19+95.29      f, ; lit.Boolean
[154] qq 1d2+c73.19+75.29+c59.39 f, ; own integer
[155] qq 2d2+c73.19+75.29+c59.39 f, ; own real
[156] qq 3d2+c73.19+75.29+c59.39 f, ; own Boolean
[157] qq 11d2+c72.19+76.29+c59.39 f, ; switch
[158] qq 8d2+c71.19+73.29+c59.39 f, ; array integer
[159] qq 9d2+c71.19+73.29+c59.39 f, ;      real
[160] qq 10d2+c71.19+73.29+c59.39 f, ;      Boolean
[161] qq 7d2+c79.19+79.29+c67.39 f, ; proc. no type
[162] qq 23d2+c79.19+79.29+c55.39 f, ;      integer
[163] qq 24d2+c79.19+79.29+c55.39 f, ;      real
[164] qq 25d2+c79.19+79.29+c55.39 f, ;      Boolean
[165] qq 1d2+c73.19+74.29+c59.39 f, ; simple integer
[166] qq 2d2+c73.19+74.29+c59.39 f, ;      real
[167] qq 3d2+c73.19+74.29+c59.39 f, ;      Boolean
[168] qq 4d2+c73.19+74.29+c59.39 f, ; label
[169] qq 14d2+c73.19+73.29+c59.39 f, ; formal array integer
[170] qq 15d2+c73.19+73.29+c59.39 f, ;      real
[171] qq 16d2+c73.19+73.29+c59.39 f, ;      Boolean
[172] qq 7d2+c73.19+79.29+c87.39 f, ; formal proc.no type
[173] qq 23d2+c73.19+79.29+c56.39 f, ;      integer
[174] qq 24d2+c73.19+79.29+c56.39 f, ;      real
[175] qq 25d2+c73.19+79.29+c56.39 f, ;      Boolean
[176] qq 1d2+c73.19+78.29+c59.39 f, ; formal simple integer
[177] qq 2d2+c73.19+78.29+c59.39 f, ;      real
[178] qq 3d2+c73.19+78.29+c59.39 f, ;      Boolean
[179] qq 4d2+c73.19+78.29+c59.39 f, ; formal label
[180] qq 11d2+c72.19+76.29+c59.39 f, ; formal switch
[181] qq 5d2+c73.19+78.29+c59.39 f, ; formal string
[182] qq 20d2+c77.19+79.29+c54.39 f, ; proc.call or ass.int.
[183] qq 21d2+c77.19+79.29+c54.39 f, ;      real
[184] qq 22d2+c77.19+79.29+c54.39 f, ;      Bool.
[185] qq      c69.19          f, ; expression
[186] qq      2                , ; end call
[187] qq 39d3+c64.19+11.29      f, ; beg exp
[188] qq 13d2+c73.19          +c87.39 f, ; undeclared
[189] qq 7d2+c11.19          +c59.39 f, ; l  sstreng
[190] qq 24d2+c11.19          +c59.39 f, ; l  st
[191] qq 6d2+c11.19          +c59.39 f, ; output no param.
[192] qq 7d2+c59.19          +c57.39 f, ; several par. no type
[193] qq 6d2+c59.19          +c57.39 f, ; several par. nonsense
[194] qq 40d3+c59.19          +c53.39 f, ; arith.in, real out
[195] qq 41d3+c59.19          +c53.39 f, ; string in, nons. out
[196] qq 42d3+c59.19          +c53.39 f, ; string in, Bool. out
[197] qq 43d3+c59.19          +c53.39 f, ; arith.in, int. out
[198] qq 44d3+c59.19          +c53.39 f, ; int.in, nonsense out
[199] qq 45d3+c59.19          +c53.39 f, ; arith.in, real out
```

```
[mode independent symbols originating in pass 3, values rel. to d50 in 3]
[Priority Action Output or operator]
```

[2.12.65]

[PASS 6, page 20, CONTROL TABLE 4 GIER]

[MODE INDEPENDENT SYMBOLS FROM PASS 4, VALUES RELATIVE TO d22]

[PRIORITY ACTION OUTPUT]

```
[ 0] qq      c89.19+  1.29      f ; end local
[ 1] qq      c96.19+104.29      f ; bypasslabel
[ 2] qq      c40.19+   8.29      f ; goto bypasslabel
[ 3] qq      c40.19+   9.29      f ; labelcolon
[ 4] qq 2.9+c49.19                ; boundcolon
[ 5] qq 2.9+c48.19+100.29        ; end bounds integer
[ 6] qq 2.9+c48.19+100.29        ; end bounds real
[ 7] qq 2.9+c48.19+100.29        ; end bounds Boolean
[ 8] qq 3.9+c44.19+ 46.29        ; thenst
[ 9] qq      c40.19+ 13.29      f ; else st
[10] qq      c46.19+ 16.29      f ; := for
[11] qq 3.9+c95.19+ 24.29        ; stepelem
[12] qq      c42.19+ 80.29+38d3.39 f ; first:=
[13] qq 3.9+c39.19+ 19.29        ; while
[14] qq 3.9+c45.19+ 20.29        ; while element
[15] qq      c68.19              f ; prepare assign
[16] qq 3.9+c39.19+ 17.29        ; simple for
[17] qq 2.9+ c6.19                ; end assign
[18] qq      c85.19              f ; do
[19] qq      c40.19+ 10.29      f ; while label
[20] qq      c38.19+ 80.29+38d3.39 f ; :=
[21] qq 2.9                      ; end ex
[22] qq 2.9+c30.19+ 38.29        ; end des
```

[MODE INDEPENDENT SYMBOLS FROM PASS 5, VALUES RELATIVE TO d10]

```
[ 0] qq      c50.19+ 81.29      f ; local switch
[ 1] qq      c50.19+ 82.29      f ; local proc
[ 2] qq      c40.19+ 83.29      f ; local label
[ 3] qq      c40.19+ 84.29      f ; blind
[ 4] qq      c51.19+103.29      f ; end pass
[ 5] qq      c82.19              f ; CR
[ 6] qq      c50.19+  6.29      f ; beg block
[ 7] qq      c50.19+  3.29      f ; beg proc
[ 8] qq      c61.19+  5.29      f ; end proc
[ 9] qq      c50.19+101.29      f ; take value integer
[10] qq      c50.19+102.29      f ; take value real
[11] qq      c50.19+102.29      f ; take value Boolean
[12] qq      c52.19+  2.29      f ; beg call
[13] qq      c40.19+  7.29      f ; end block
[14] qq      c76.19+ 98.29      f ; bounds
[15] qq      c88.19+  0.29      f ; beg local
```


[i=2d3

ACTION PRIORITY CHECK OUTPUT OUTPUTSYMBOL]

```

[ 2] qq c14+ 2.19+c25.29          ; prep. undecl. ass.
[ 3] qq c14+ 3.19+c22.29          ; undecl. 3
[ 4] qq c14+ 4.19+c22.29          ; undecl. 4
[ 5] qq c14+ 5.19+c22.29          ; undecl. 5
[ 6] qq c14+ 6.19+c22.29          ; undecl. 6
[ 7] qq c14+ 7.19+c22.29          ; undecl. 7
[ 8] qq c14+ 8.19+c22.29          ; undecl. 8
[ 9] qq c14+ 9.19+c22.29          ; undecl. 9
[10] qq c14+10.19+c22.29          ; undecl. 10
[11] qq c14+11.19+c22.29          ; undecl. 11
[12] qq c13+ 9.19+c16.29+ 51.39 ; i+ : +
[13] qq c13+ 9.19+c16.29+ 52.39 ; i- : -
[14] qq c13+10.19+c16.29+ 53.39 ; ix : x
[15] qq c13+ 9.19+c16.29+ 35.39 ; neg: neg
[16] qq c14+ 9.19+c16.29          ; pos
[17] qq c13+ 9.19+c17.29+ 51.39 ; r+ : +
[18] qq c13+ 9.19+c17.29+ 52.39 ; r- : -
[19] qq c13+10.19+c17.29+ 53.39 ; rx : x
[20] qq c13+10.19+c17.29+ 54.39 ; /: /
[21] qq c12+11.19+c17.29+ 57.39 ; r↑ : ↑ real
[22] qq c13+10.19+c18.29+ 55.39 ; ⋮ : ⋮
[23] qq c12+11.19+c16.29+ 57.39 ; i↑ : ↑real
[24] qq c14+ 8.19+c20.29+ 64.39 ; < : <
[25] qq c14+ 8.19+c20.29+ 65.39 ; ≤ : ≤
[26] qq c14+ 8.19+c20.29+ 66.39 ; = : =
[27] qq c14+ 8.19+c20.29+ 67.39 ; ≥ : ≥
[28] qq c14+ 8.19+c20.29+ 68.39 ; > : >
[29] qq c14+ 8.19+c20.29+ 69.39 ; ≠ : ≠
[30] qq c13+ 6.19+c21.29+ 70.39 ; ^ : ^
[31] qq c13+ 5.19+c21.29+ 71.39 ; v : v
[32] qq c13+ 3.19+c21.29+ 72.39 ; ≡ : ≡
[33] qq c13+ 7.19+c21.29+d20.39 ; -, : -,
[34] qq c14+ 2.19+c24.29+d28.39 ; prep. int. ass. : prepare assign
[35] qq c13+ 2.19+c23.29+d28.39 ; prep.real ass. : prepare assign
[36] qq c13+ 2.19+c19.29+d28.39 ; prep. Bool. ass. : prepare assign
[37] qq      1.19                  ; (
[38] qq c13+ 2.19+ c3.29+ 49.39    ; := : :=
[39] qq c14+ 2.19+c29.29[RESULT TYPE] ; actual parameter
[40] qq c10+ 2.19+c27.29+2d2.39 ; arith.in, val. out: real
[41] qq c10+ 2.19+c28.29+6d2.39 ; string in,nons. out: nonsense
[42] qq c10+ 2.19+c28.29+3d2.39 ; string in.Bool. out: Boolean
[43] qq c10+ 2.19+c27.29+1d2.39 ; arith.in, int. out: integer
[44] qq c26+ 2.19+c27.29+6d2.39 ; int.in, nons. out: nonsense
[45] qq c10+ 2.19+c27.29+2d2.39 ; arith.in, val. out: real
[46] qq c10+ 2.19+c93.29+1d2.39 ; array in, int. out: integer
[47] qq c26+ 2.19+c27.29+7d2.39 ; int. in, no type: no type
[48] qq c10+ 2.19+c100.29+2d2.39 ; bool in, real out: real

```

d7:

[TABLE HAS 47 WORDS, FROM 2d3 TO 48d3]

e ; end GIER table drum block

d i=d2

```
[ 0] qq 367.9+679.19+143.29+1.32+1.38 ; undeclared
[ 1] qq 434.9+ 42.19+312.29+1.32+1.38 ; integer
[ 2] qq 338.9+ 49.19+408.29+1.32+1.38 ; real
[ 3] qq   6.9+348.19+516.29      +2.38 ; boolean
[ 4] qq   1.9+ 96.19+642.29      +3.38 ; label
[ 5] qq          64.19+769.29      +4.38 ; string
[ 6] qq          544.19+904.29      +1.38 ; nonsense
[ 7] qq          512.19              ; no type proc.
[ 8] qq          128.19              +4.32+1.35 ; int. arr
[ 9] qq          128.19              +4.32+2.35 ; real arr.
[10] qq          128.19              +4.32+3.35 ; bool. arr.
[11] qq          128.19              +6.32+4.35 ; switch
[12] qq              +127.29          ; unknown type
[13] qq 367.9+679.19+143.29+1.32+1.38 ; undeclared
[14] qq          128.19              +1.35 ; for.int.arr.
[15] qq          128.19              +2.35 ; for.real arr.
[16] qq          128.19              +3.35 ; for.bool.arr.
[17] qq           8.19                ; proc 1 = procedure with
[18] qq          16.19                ; parameters, call and assign
[19] qq          24.19                ;
[20] qq 434.9+552.19+312.29          +1.38 ; proc 2 = procedure without
[21] qq 338.9+560.19+408.29          +1.38 ; parameters, call and assign
[22] qq   6.9+856.19+516.29          +2.38 ;
[23] qq 434.9+544.19+312.29          +1.38 ; proc 3 = procedure without
[24] qq 338.9+544.19+408.29          +1.38 ; parameters, call only
[25] qq   6.9+832.19+516.29          +2.38 ;
d9:
```

```
d i=39d29, d4= d5+50 ;
d e18=c86, e16=k-e70 ; Pass word parameters
```

```
e ;
```

```
d i=4e21, e22=e16-e15-1 ;
qqe15.9-e24.9+e18.19+e22.36+e22.34+205e13.39; Pass word
d e15=e16 ;
s
```

```

b k=e15 + e70, i = 205e13, a30, b20, c50, d12          ;
d e17 = i                                              ;
d d10 = 265 [output base] , d=146 [lit base]          ;
d d1 = 108 [max interest], d4 = 640 [value allowed]    ;
d d5 = 768 [take value], d6 = 3d10 [local blind, output] ;
d d7 = 20 [proc call and assign constant]              ;
d d8 = 188 [undeclared, output], d11=9d10 [value integer];

c44:hs c40 , ga b15 ; start pass 5: min identifier:= takebyte;
    sc b8      ITA ; maxp:= basestack - min identifier;
    pa b8      t 512 NTA ; if maxp < -512 then maxp:= - 512;
    hs c40 , ga b11 ; own address:= take byte;
    hs c40 , hs e3 ; output(take byte); comment stackreference for block 0
    pm r1 , ud 5e4 ; relative to display;
    bs s ; for i:= ,lastavailable, step -1 until
    gm 40e25 V -1 MC ; laststandardidentifier do
    grn (r-1) t -1 M ; storewithmarks(,blob,, 3, decltable[i]);
    it (b15) , bs (r-2) ; for i:= i - 1 step -1 until minidentifier do
    ps s-1 , hv r-4 ; storewithmarks(0,0, decltable[i]);
    pp 0 , hv c41 ; ds:= ,firstavailable, - 1; blocknumber := 0;
    ; go to from sstart;
b4: qq ; decl
b5: qq ; number byte
b13:qq 1.32-1.34 ;
b16:qq d8.34 ; undeclared
b18:qqf d7.9 ; constant for proc value
b19:qq 1.2-11.34 ; constant for take value
c40:pm (e1) Xt 1 ; integer procedure take byte;
    hs e2 LA ; takebyte:= input;
    hh s ;
c0: ;
a2: qq (e4) t 1 ; CARRET: CRcounter:= CRcounter + 1;
c22: ;
a9: pmn(e1) X 1 ; next 1: byte:= input;
    hs e2 LA ; byteread: if byte > 512 then go to outputdescription;
a5: ga r2 ; if byte > ,ofinterest, then
    hv a1 LT ;
b17:bs t d1 ;
a8: ps a2 , hv e3 ; next: begin output(byte); go to next 1 end;
    ck -2 , ga r1 ;
    pmn[byte:4]Xtd2 IRC ; R:= table[byte:4]; marks:= marksoftable[byte:4];
    ga b1 , gt b2 ; output:= part 1(R); actionnumber:= part 2(R);
    hv a3 NC ; if marks ≠ ,notdecl, then
    tk 20 , ar (e1) ; begin decl:= (part 3(R) + byte)×2↑5 + part 4(R)×2↑35;
    ck 15 ; if marks ≠ ,nblock, then setpart 1(decl)
    ; to: (blocknumber);
b3: ar[block no] D LRB ; if marks = ,number, then
    gr b4 ; begin numberbyte:= take byte;
    hv a4 LRA ; decl:= decl + numberbyte×2↑14
    hs c40 , ga b5 ; end;
    ck -16 , ac b4 ;

```

```

a4: hs c40 , ga b6 ; next identifier: R:= byte 1:= take byte;
    hv a22 LZ ; if R = 0 then go to CARRET 2;
    hv a5 NT ; if byte < 512 then go to byteread;
    hs c40 , ga b7 ; declCR:= takebyte;
b6: pmn[byte1]X IRC ; R:= decltable[byte 1];
    hh b2 LT ; marks R:= marksof(decltable[byte 1]);
    hv b1 LZ ; if part1(R)>512 then go to action[actionnumber];
    ca (b3) NB ; if R ≠ 0 then
    hv c31 NB ; begin if part 1(R) = blocknumber then
    pp p1 , ck -5 ; begin if marksR≠,standard, then go to error1 end;
b8: bs pd3 , hv c32 ; ds:=ds+1; if ds>minidentifier then go to error 2;
    ar (b6) D ; storewithmarks(bits(5,34,R)×2↑(-5) +
    ; bytex2↑30,marksR,stack[ds]);
    gr pd3 MRC ; information 1:= information 1 + 1;
    qq (3e4) t 1 ; storewithmarks(0, notfirst, decltable[byte 1]);
    grn(b6) MA ; end stack previous declaration for same
a3: ; identifier
b1: pmn[outbyte] DX ; end prepare declaration of one identifier;
    hs e3 NZ ; a3: if outbyte ≠ 0 then output(outbyte);
b2h:pan b1 ,hv[action] ; go to action[actionnumber];
a22:qq (e4) t 1 ; CARRET 2 : CRcounter:= CRcounter + 1;
    pm 5d10 XD ; output(CARRET);
    ps a4-1 , hv e3 ; go to next identifier;
c4: qq (b3) t 1 ; begin: blocknumber:= blocknumber + 1;
    pm b9 , pp p2 ; ds:= ds + 2; stack[ds]:= local address;
    gm pd9 , pa b9 ; localaddress:= 0;
c41:hs c40 , hs e3 ; fromstart: output(takebyte);
    ; comment base for working storage;
    hs c40 , ga b10 ; variableaddress:= takebyte; formaladdress:= 2;
    pa b12 t 2 IQB ;
    bs (b8) , hv c32 ; ds:= ds + 1;
    grn pd3 , hv a9 ; if ds > minidentifier then go to error 2;
    ; declstack[ds]:= 0; intypeproc:= false;
c26: ; go to next 1;
b15:ps[min ident], ps s1 ; end: i:= minidentifier;
    pm s X ; cleartable: i:= i + 1;
    hv a10 LB ; if marksof(decltable[i]) ≠ ,standard, then
    nc (b3) NA ; begin if part 1[R] = blocknumber then
    hh c26 ; decltable[i]:= 0; go to cleartable
    grn s , hh c26 ; end;
a10:qq (b3) t -1 ; blocknumber:= blocknumber - 1;
    arn pd3 , pp p-1 ; unstack: R:= declstack[ds];
    ; marks R:= marksof(declstack[ds]);
    hv a20 LZ ; ds:= ds - 1; if R = 0 then go to a20;
    ga r2 , tk 10 ; storewithmarks(bits(10,39,R)×2↑5, marksR,
    ck -5 IRC ; decltable[bits(0,9,R)]; go to unstack;
    gr MRC ;
    hv r-5 ;
a20:pm pd3 , gm b9 ; a20: local address:= stack[ds];
    pp p-1 , hv a9 ; ds:= ds - 1; go to next 1;
c25:hs c40 , mt r-1 ; bounds: output(localaddress - takebyte);
    ar (b9) D ; begcall: output(takebyte); go to next 1;
    hs e3 ;
c23:hs c40 , hv a8 ;

```

```

c24:arn d6      D      LQB      ; beg local: if intypeproc then output(localblind);
      hs e3      LQB      ;
      can(b14) , hv a9      ; if counter = 0 then go to next 1;
      it (b15) , pa b6      ; for i:= min identifier + 1 step 1 until topid do
a13:arn(b6) t 1      ; if store[i] < 0 then error 7;
      hv c37      LT      ; counter:= 0;
a15:bs (b6) t 38 e25 ; go to next 1;
      pa b14 , hv a9      ;
      hv a13      ;
c1: pm (b17) DX d      ; literal: output(byte + litbase);
      hs e3      ; for i:= 1 step 1 until 4 do output(takebyte);
      hs c40 , hs e3      ; go to next 1;
      hs c40 , hs e3      ;
      hs c40 , hs e3      ;
      hs c40 , hs e3      ; only CDC
      d i=i+e69-1      ; CDC: 5 bytes
      hs c40 , hv a8      ;
c8: nt (b5) , qq (b9) ; switch: localaddress:= localaddress - numberbyte;
c10:arn b5 , hs e3      ; procedure: output(numberbyte);
c12:      ;
b9: ps[local address]t -1 ; local: s:= localaddress:0 localaddress - 1;
a6: pm b4 , gm (b6) ; storedcl: decltable[byte 1]:= decl1;
a11:it s , pt (b6) ; setrelative: setpart 2(decltable[byte 2]) to: (s);
      hv a4      ; go to nextidentifier;
c9: pt b2 t c11      ; array: variableaddress:= variableaddress - (numberbyte+1);
      nt (b5) t 2      ; actionnumber:= actionnumber + 1;
c11:      ;
b10:ps[var adr.] Vt -1 ; variable: s:= variableaddress:= variableaddress - 1;
      ; go to storedcl;
c13:      ;
b11:ps[own adr.] t -1 ; own: s:= ownaddress:= ownaddress - 1; go to storedcl;
      hv a6      ;
c6: arn pd3 , tk 10 ; proc value: local address:= local address - 1;
      ck 20 , ud c12 ; intypeproc:= true; decltable[byte 1]:=
      ca d8 , sr b18 ; ((if bits(31,39,declstack[ds]) = undeclared then 0
      ar b18 IQB ; else proc call and assign const) +
      ck 15 ; bits(10,39,declstack[ds]))x215; go to next 1;
      gr (b6) , hv a9 ;
c20:nc d4 , hv c36 ; value: if part 1(R) ≠ ,allowed, then go to error 6;
      ar b19 , gr (b6) ; decltable[byte1]:= R + ,constantforvalue,;
      hv a4      ; go to nextidentifier;
c14:hv c33 LT      ; specify: if part 1(R) ≥ 512 then go to error 3;
b14:qqn[counter] t -1 ; counter:= counter + 1; decltable[byte 1]:= decl1;
      pm b4 , gm (b6) ; go to nextidentifier;
      hv a4      ;

```

```

c21: hv c34 NT ; formal: if part 1(R) < 512 then go to error 4;
      qq (b14) t 1 ; counter:= counter + 1;
b12: ps[formal adr] t 1 ; s:= formaladdress:= formaladdress + 1;
      nc d5 , hv a12 ; if part 1(R) = ,takevalue, then
      tk -5 , tk 30 ; begin
      ar d11-165 D ; output(bits(25, 34, R) - 165 + value integer;
      hs e3 ; output(formaladdress - 2);
      arn s-2 D ; end;
      hs e3 ; setformal: setpart 1(decltable[byte 1]) to: (blocknumber)
a12: it s ; set part 2(decltable[byte 1], formal address);
      pt (b6) , ps (b3) ; go to next identifier;
      gs (b6) , hv a4 ;
a1: pm (b17) ; outputdescription: R:= decltable[byte1];
      hv a8 LC ; marks R:= marksof(decltable[byte1]);
      cln -14 ; if marks R = standard then go to next;
      hv c35 LZ ; if R = 0 then go to error 5;
      ck -1 , hs e3 ; information 2:= information 2 + 1;
      cln -6 , ck -4 ; output(bits(27,34,R));
      hs e3 ; output(bits(3 ,25,R));
      cln -10 , hs e3 ; output(part 2(R));
      qq (2e4) t 1 ; output(part 1(R));
      cln -10 , hv a8 ; go to next 1;
c31: hs c43 , qq s+e45 ; error 1: m:= {<+declar.>}; go to c43;
c32: ps c35 , hv e5 ; error 2: message({<stack>}); go to initialize transl;
c33: hs c42 , qq s+e46 ; error 3: m:= {<+specific>}; goto c42;
c34: hs c42 , qq s+e48 ; error 4: m:= {<-specific.>}; go to c42;
c35: pm b16 , gm (b17) ; error 5: byte:= undeclared;
      hs e5 , qq e38 ; message({<-declar>});
      hv a1 , qq s+e47 ; go to output description;
c36: hs c42 , qq s+e49 ; error 6: m:= {<value>}; go to c42;
c37: hs c42 , qq s+e50 ; error 7: m:= {<-formal>};
c42: bs (b14) , hv a14 ; c42:
      pa b14 t 500 ; if counter ≤ 0 then
c43: arn s , gt b20 ; begin counter := 500;
      it (e4) , pt b ; c43: b:= CR counter;
b7: it -1 , pa e4 ; CRcounter:= declCR;
      hs e5 ; message(m);
b20: qq , qqs ;
b: pa e4 ; CRcounter:= b; end;
a14: pm b16 , gm (b6) ; byte 1:= undeclared;
      can s-c37 , hv a15 ; if error 7 then go to a15;
      hv a4 ; go to next identifier;
d c7=e92 ; define end pass

```

```
d2: [0] qq 5d10.9 + c0.19 ; CARRET
[1] qq c1.19 ; constant
[2] qq ;
[3] qq ;
[4] qq 6d10.9 + c4.19 ; begin block
[5] qq 7d10.9 + c4.19 ; begin procedure
[6] qqf c6.19 + 157.29 , ; proc value
[7] qq 4d10.9 + c7.19 ; end pass
[8] qqf d10.9 + c8.19 + 125.29 ; decl switch
[9] qqf c9.19 + 121.29 ; decl array
[10] qqf 1d10.9 + c10.19 + 121.29 ; decl procedure
[11] qqf c11.19 + 120.29 , ; decl simple
[12] qqf 2d10.9 + c12.19 + 120.29 , ; decl label
[13] qqf c13.19 + 101.29 , ; decl own
[14] qq c14.19 + 124.29 + 4.37 , ; specify switch
[15] qq c14.19 + 108.29 + 4.37 , ; specify array
[16] qq c14.19 + 108.29 + 4.37 , ; specify procedure
[17] qq c14.19 + 107.29 + 5.37 , ; specify simple
[18] qq c14.19 + 107.29 + 4.37 , ; specify label
[19] qq c14.19 + 105.29 + 4.37 , ; specify string
[20] qq c20.19 , ; value
[21] qqf c21.19 , ; formal
[22] qq c22.19 ; switch list
[23] qq 12d10.9 + c23.19 ; begin call
[24] qq 15d10.9 + c24.19 ; begin local
[25] qq 14d10.9 + c25.19 ; bounds
[26] qq 13d10.9 + c26.19 ; end block
[27] qq 8d10.9 + c26.19 ; end procedure
d3: qq ;

d d9=d3-1 [stack start - 1] ;

d e18=c44 [start pass] ; information
d i=39i, e16=k-e70 ; for pass word
e ;

d i=3e21, e22=e16-e15-1 ; Load pass word
qq e15.9-e24.9+e18.19+e22.36+e22.34+205e13.39,,;

d e15=e16 ; set first free track

s
```

[27. jan. 65]

[Pass 4, page 1]

b k = e15 + e70, i = 205e13, a30, b16, c40, d30

d e17 = i

[Input byte values and functions of them]

d d = 109 [bounds int], d1 = 168 [of interest]

d d4 = 12 [max literal], d6 = 479 [511 - min expres byte]

d d11 = 41 [min type proc],

[Output byte values]

d d22 = 242 [output base for processing in pass 6]

d d3 = 2d22 [go to bypass label], d7 = 185 [expression]

d d8 = d22 [end local], d9 = 1d22 [bypass label]

d d12 = 24 [procvalue], d13 = 19d22[while label]

d d14 = 15d22 [prepare assign], d15 = 28 [end pass]

d d16 = 96 [beg local], d17 = 187 [beg express]

b1: qq d15 ; out byte, initially end pass

b2: qq ; counter

b9: qq ; last identifier

b10:qq 1.7-1.23-1.28-1.34 ; trouble word

b11:qq ; beg end word

b14:qq ; working location

c30:pm (e1) X 1 ; integer procedure take byte;

hs e2 LA ; begin

hr s1 NZ ; again: byte:= take byte:= input; if byte = ,CR, then

hs e3 ; begin output(,CRout,);

qq (e4) t -1 ; CRcounter:= CRcounter - 1; go to again end

; end takebyte;

c26:hv c30 , ps a3 ; procedure c26; begin c27; goto byteread end;

c27:pm d16 DX ITB ; procedure c27; begin b14:= R; inblock:= false;

gm b14 , hv e3 ; output(beg local)

c28:hv s1 LZB ; end;

pmn d9 DX IZB ; procedure c28; if set bypass label then

hv e3 ; begin set bypass label:= false; output(bypasslabel) en

c29:hv s1 LZA ; procedure c29; if -, operand taken then output(beg ex)

pm d17 XD ;

hv e3 ;


```

c32:hs  c30      ; procedure enterlist(counting);
    hr  s1      NT      ; begin
    pp  p1      , ud  c  ; nextidentifier: R:= takebyte; if R > 512 then
    hs  c31      ; begin p:= p + 1; if p > maxp then stackoverflow;
    gr  pd2      MA      ; storewithmarks(byte×2↑30 + CRcounter×2↑20,
    it (e4)      , pt  pd2 ; ,withcounter,, stack[p]);
    qq (s1)      t    1   ; counting:= counting + 1; go to nextidentifier
    hv  c32      ; end
                        ; end enterlist;

c31:it p-512    , pt  b16 ; procedure stackoverflow;
    gp  2e4      ; if p > ,absmaxp, then alarm(‡<stack‡, ,stop,)
    bs p-512    t d21    ; else maxp:= maxp + ,deltamaxp,;
    hs  e5      ;
    hr  s1      , qqn e38 ;

c36:                ; start pass 4: inswitch:= warning:= false;
    arn(e1)      t    1 IQC ; last identifier:= input; p:= 0;
    ga  b9      , pp  0   ; take byte; input byte:= end block;
    hs  c30      ;
    pa (e1)      t    100 ;
a1:  arn b1      , hs  e3   ; outandbyteread: output(outbyte); go to byteread;
a2h: hv  a4      , pm  b1   ; out: byte:= outbyte;
    hs  e3      X        ; next: output(byte);
a3:  pm (e1)      XV    1   ; next 1: byte:= input;
a4:  pm (e1)      XV      ; byteread: if byte > ,ofinterest, then go to next;
    hs  e2      LA      ; R:= table[byte:4]; marks R:= marksof(table[byte:4]);
    ga  a5      ; actionnumber:= part2(R); outbyte:= part1(R);
a5:  bs [byte] t d1 NT    ; if marks R ‡ ,nostacking, then
    ps a3-1      , hv  e3   ; begin p:= p + 1; if p > maxp then stackoverflow;
    ck -2        , ga  a6   ; storewithmarks(if marks R = ,stop, then R×2↑20 else
a6:  pmn[byte:2]Xt b IRC ; (byte×2↑30 + (if marks R = ,withcounter, then
    ga  b1      , gt  a7   ; counter×2↑20 else 0)), marks R, stack[p])
    hh  a7      NC      ; end;
    pp  p1      ; go to action[actionnumber];
c:b16:bs p-512 t - 512 ;
    hs  c31      ;
    tk  20      V    LC    ;
    pm (e1)      X      ;
    gr  pd2      MRC     ;
    qq          V    LRB   ;
    it (b2)      , pt  pd2 ;
a7h:                , hh[action];

```

```

c1: qq (e4)    V    -1    ; CR: CRcounter:= CRcounter - 1; go to out;
c2: qq (b2)    t     1    ; boundop1: counter:= counter + 1; go to out;
c3h:hh  a2      , pa  b2    ; bounds: counter:= 0;
      hv  r3      NZB      ;   if -, set bypass label then
      arn d3     D     IZB   ;   begin set bypass label:= true;
      hs  e3      ;         output(goto bypass label);
      it (a5)    , qq (b1)   ;   end; outbyte:= outbyte + byte; go to out;
c4: hh  a2      , it (a5)   ; literal: outbyte:= byte;
      pa  b1      ;         for i:= 1 step 1 until if CDC then 5 else 4
a29:pa  a8      t e69+4     ;         do output(input); go to out and byte read;
      pm (e1)    X     1     ;
      hs  e2      LA        ;
a8: bt          t     -1    ;
      ps  a29     , hv  e3    ;
c5: hv  a1      , hs  c30    ; trouble: a9:= take byte : 4;
      ck  -2      , ga  a9    ;   if a9 < 2 then
      pa  b15     , it  2     ;   begin for i:= 1 step 1 until if CDC then 5 else 4
      bs (a9)    , hv  a9     ;   do input; go to trouble
a30:pmn(e1)    t     1     ;   end;
      hs  e2      LA        ;   if a9 > 39 then go to trouble;
b15:bt  0       t     350    ;   if bits (a9, a9, troubleword) = 1 then
      d i=i-e69   ;   go to byteread;
      bt  0       t     260   ;   go to trouble;
      d i=i+e69-1 ;
      hh  c5      ;
      hv  a30     ;
a9: bs          t 39 NT      ;
      hh  c5      ;
      arn b10     , ck (a9)   ;
      hv  a4      LO        ;
c6: hh  c5      , hs  c29    ; parameter: if -, operandtaken then output(,begex,);
a10:qqn(b2)    t 1  IZA     ; takeparameter: operandtaken:= false;
      pp  p1      , hs  c30   ;   counter:= counter + 1;

```

```

    gr pd2      M      ; storewithmarks(take byte×2↑30, 0, stack[p]);
    bs (pd2) t d4      ; if bits(0, 9, stack[p]) > maxliteral then
    hv a11      IZA     ; begin operand taken:= false; goto expression end;
    hh a12      LT      ; if bits(0, 9, stack[p]) < 512 then
a13:pm (e1) X 1      ; begin for i:= 1 step 1 until if CDC then 5 else 4
    hs e2      LA      ; do begin storewith marks(input×2↑30,
    gr pd18     MB      ; 1, stack[p+4]); p:= p - 1;
                        ; comment s = c30 since hr-jump returning to 2a10
    pp p-1     , ps s-1 ; end;
    bs sd5     , hv a13 ; p:= p + (if CDC then 10 else 8);
a12h:pp pe69+e69+8,hs c30 ; if takebyte > this begins expr then
    ga a14     , it d6  ; expression: begin if in switch then c28;
a14:bt      , hvn a15 ; output(outbyte);
a11:hs c28    LQB      ; a16: R:= stack[p];
    arn b1     , hs e3  ; in switch:= mark b(stack[p]);
a16:pmn pd2 X IQB      ; if operand taken then output(R);
    hs e3      LZA     ; if -, in switch then stack[p]:= expr;
    pa pd2     V d7 NQB ; p:= p - 1; goto a16;
    pp p-1     , hv a16 ; end;
a15:ud c      ; if p > maxp then stackoverflow;
    hs c31     ; operand taken:= false;
    hv a4      IZA     ; goto byteread;
c33:pm d13 DV   ; forelem: if warning then
                        ; begin warning:= false; output(,whilelabel,) end;
                        ; go to out;
c34:pm d14 D    ; assign: if warning then
    hs e3 X LQA   ; begin warning:= false; output(,prepare assign,) end;
c37:pm r V IQA   ; go to out;
c35:pm a23 IQA   ; setwarning: warning:= true; go to out;
c7: hh a2 , hs c29 ; declareswitch: c29;
    arn b2 , ac b4 ; numberoflocals:= numberoflocals + counter;
                        ; inswitch:=false;
    hs c32 IQB     ; enterlist(numberoflocals); go to byteread;
    qq (b4) , hv a4 ;
c8: it (b2) t 1   ; declarearray: numberofvariables:=
                        ; numberofvariables+counter;
    qq (b5) , hs c32 ; a17:= 0; counter:= counter + 1; enterlist(a17);
a17:arn 0 D      ; output(a17); output(numberofvariables);
    pm b5 , ac b5 ; numberofvariables:= numberofvariables + a17;
    pa a17 , hs e3 ; go to outandbyteread;
    hs e3 X      ;
c9: hv a1 , pp p-1 ; declareproc: p:= p - 1;
    hv a24 NTB    ; if -, inblock then go to proc unstack;
a20:hs c28 NZB   ; unstack decl: if -, bypasslabel then c28;
    pm d8 DX      ; output(end local); go to procunstack;
c10:hh a19 , hs c32 ; declaresimple: enterlist(numberofvariables);
b5: qq [number of variables]; goto byteread;
c11:hv a4 , hs c32 ; declarelabel: enterlist(numberoflocals);
b4: qq[number of locals] ; go to outandbyteread;
c12:hv a1 , hs c32 ; declareown: enterlist(numberofowns); go to byteread;
b3: qq [number of owns] ;
    hv a4      ;

```

```

c13:hv a20          LTB      ; begin: if inblock then go to unstack decl;
      arn b11      , ac  b11  ;      beg end word:= 2xbeg end word; go to set in block;
c14:hv a21          , hs  c29  ; begin call: c29; operand taken:= false;
      hs  c30          IZA    ;      identifier:= takebyte;
      ga  b12          ;      output(outbyte);
a19:arn b1          , hs  e3    ; procunstack: addrofprocdecl:= p;
a24:gp a22          , arn pd2   ; unstack: R:= stack[p]; marks R:= marksof(stack[p]);
      pp  p-1        , ga  b1   ;      p:=p-1; outbyte:= part 1(R); numberbyte:= part 2(R);
      hv  a23          LC      ;      if marks R:= ,stop, then go to action[numberbyte];
      qq          V      NA    ;      if marks R = ,withcounter, then output(numberbyte);
      tk  10          , hs  e3   ;      output(outbyte);
      arn b1          , hs  e3   ;      go to unstack;
c15:hh a24          , arn b11   ; endclean: begendword:= begendword_2;
      ck  -1          , gr  b11  ;
a21:pm b11          ITB      ; set in block: inblock:= bit(0, 0, beg end word) = 1;
                                ;      go to next 1;
c16:hv a3           , hs  c28   ; endproc: c28;
      hs  c30          ;      output(take byte);
      hs  e3           ;
c17:qq (b13) t 1 IZB      ; endblock: blocknumber:= blocknumber + 1;
      it (3e4) , bs (b13) ;      setbypasslabel:= true;
      qq (3e4) t 1       ;      if blocknumber > maxblocknumber then
      arn b5          , ck  10   ;      maxblocknumber:= maxblocknumber + 1;
      ar  b4          , ck  10   ;      setpart 3(stack[p]) to: (numberofvariables);
      ac  pd2         , pa  b4   ;      setpart 4(stack[p]) to: numberoflocals);
      arn b11         , ck  -1   ;      inblock:= true; number of locals:= 0;
      ar  512 D       ITB      ;      begendword:= begendword_2 + 239;
      gr  b11         , pa  b5   ;      numberofvariables:= 0; go to out;
c18:hh a2           , arn b2    ; endcall: setpart 4(stack[p]) to: (counter);
      ck  10          , ac  pd2  ;      output(,bypasslabel,); counter:= 0;
      pm  d9          DV       ;      go to takeparameter;
c19:pmf r           V      IQB  ; endswitch: inswitch:= true;
      hs  e3          X        ;      counter:= 0; go to takeparameter;
      pa  b2          , hv  a10  ;
c20:hh c9           LTB      ; specify: if inblock then go to declare proc;
      pa  b2          , hs  c32  ;      counter:= 0;
      qq          , hv  a4      ;      enterlist(blind); go to byteread;
a23:gt r , hh[stackaction];

```

```

c21: hv a20 LTB ; formal: if inblock then go to unstackdecl;
a28: pa b2 , hs c30 ; counter:= 0;
    hv a1 NT ; moreformals: R:= takebyte;
    qq (b2) t 1 ; if R < 512 then go to outandbyteread;
    pm (e4) DX ; counter:= counter + 1;
    hs e3 ; output(CRcounter); output(R);
    hs e3 X ; go to moreformals;
c22: hh a28 , tk 30 ; outofcall: output(identifier); output(counter);
b12: pm DX ; counter:= part 4(R); go to out;
    hs e3 ;
    arn b2 , gm b2 ;
    hs e3 ;
c23: hh a2 , hs c27 ; outofblock: c27; setbypasslabel:= true;
a25: qq V IZB ; blockoutput:
    qq (b4) t 1 ; output(-numberoflocals);
a27: srn b4 , hs e3 ; output(-numberoflocals - numberofvariables);
    srn b4 , sr b5 ; if blocknumber < 0 then
    pm b14 , hs e3 ; begin
    bs (b13) , hs a26 ; endpass 4: output(-maxblocknumber-numberofowns + 1021);
    srn(b3) D 2 ; output(numberofowns + 2);
    sr (3e4) D 1 ; output(lastidentifier);
    pm b3 , hs e3 ; go to endpass;
    hs e3 X ; end;
    arn b9 , hs e3 ; numberoflocals:= part 4(b14);
    hv e92 ; numberofvariables:= part 3(b14);
a26: cl -10 , ga b4 ; blocknumber:= blocknumber - 1;
    cl -10 , ga b5 ; begendword:= begendword*2 - begendword:2↑38×2↑40;
b13: qq -1[blockno]t -1 ; inblock:= begendword ≥ 2↑39;
    arn b11 , ac b11 ; if setbypasslabel then go to out;
    pm b11 ITB ; number of locals:= number of locals + 1;
    hh a2 NZB ; go to out and byte read;
    qq (b4) t 1 ;
c24: hv a1 , pp p1 ; outofproc: p:= p + 1;
a22: pm 0[addr of proc decl] V d20 NTB; if inblock then begin counter:= 0; c26 end;
    pa b2 , hh c26 ; storewithmarks(stack[adresofofprocdecl],
    gm pd2 MA ; ,withcounter,, stack[p]);
    gr b14 , hs c32 ; b14:= R; enterlist(blind);
    qq 0 , it (pd19) ; if part 1[table[p-1]] ≥ ,mintypeprocedure, then
    bs d11 , hv a27 ; begin output(CRcounter); output(part 1(table[p]));
    pm (e4) DX ; output(,procedure,);
    hs e3 ; numberoflocals:= numberoflocals + 1
    pm (pd2) DX ; end;
    hs e3 ;
    pm d12 DX ; go to blockoutput;
    ps a25 , hv e3 ;

```

[Pass 4, page 7, Control table.

Input bytes smaller than the limit OF INTEREST initiate an action controlled by the following table, which is entered with the argument input byte $_4$. The actions starts with a stacking process depending on the marks of the control word:

Marks	Put in stack	Control word bits
0	Nothing	
A	, input byte $\times 2^{\uparrow 30}$ + counter $\times 2^{\uparrow 20}$,
B	f input byte $\times 2^{\uparrow 30}$	f
C	f, output $\times 2^{\uparrow 30}$ + action $\times 2^{\uparrow 20}$	f,

where output and action are taken from bits 20 to 39 of the control word.

After the stacking the algorithm jumps to the action given in bits 10-19 with the parameter given in bits 0-9 assigned to outbyte]

[output, action, stackoutput, action]

```

[0] b:qq          c1.19          ; CARRET: CRout, CR
[4+] qq          4.9+c4.19       ; literal <type>: -, literal
[8]  qq          8.9+c4.19       ; (not used)
[12] qq          12.9+c4.19      ; (not used)
[16] qq          22d22.9+c6.19    ; switchparam: enddesex, parameter
[20] qq          21d22.9+c6.19    ; callparam: endex, parameter
[24] qq          186.9+c14.19     ; begcall: endcall, begcall
[28] qq          186.9+c14.19     ; begfunc: endfunc, begcall
[32] qq          187.9+c7.19,     ; declswitch: begex, declare switch
[36+] qq          100.9+c8.19,    ; declarray <type>: bounds, declare array
[40+] qq          c9.19,         ; declproc <type>: -, declare proc
[44+] qqf         c10.19         ; declsimple <type>: -, declare simple
[48] qqf         3d22.9+c11.19    ; decllabel: labelcolon, declare label
[52+] qqf         c12.19         ; declown <type>: -, declare own
[56] qqf         c20.19         ; specsswitch: -, specify
[60+] qqf         c20.19         ; specarray <type>: -, specify
[64+] qqf         c20.19         ; specproc <type>: -, specify
[68+] qqf         c20.19         ; specsimple <type>: -, specify
[72] qqf         c20.19         ; speclabel: -, specify
[76] qqf         c20.19         ; specstring: -, specify
[80] qqf         c20.19         ; value: -, specify
[84] qq          84.9+c21.19      ; formal: formal, formal
[88] qqf         22d22.9+c19.19    ; switchlist: enddesex, endswitch
[92] qqf         21d22.9+c18.19+92.29+c22.39, ; endcall: endex, endcall, begcall, out of cal
[96] qq          c15.19         ; endclean: -, endclean
[100] qqf        104.9+c17.19+16.29+c23.39, ; endblock: endblock, endblock, begblock, outo
[104] qqf        108.9+c16.19+20.29+c24.39, ; endproc: endproc, endproc, begproc, outofpro
[108] qq          5d22.9-d.9+c3.19 ; bounds: endbounds, bounds
[112] qq          4d22.9+c2.19    ; boundcolon: boundcolon, boundcolon
[116] qq          c13.19         ; begin, -, begin
[120] qq          a3.19          ; i: -, next 1
[124] qq          18d22.9+a2.19    ; do: do, out
[128] qq          8d22.9+a2.19    ; thenst: thenst, out
[132] qq          9d22.9+a2.19    ; elsest: elsest, out
[136] qq          c5.19          ; trouble: -, trouble
[140] qq          10d22.9+c33.19   ; := for: := for, forelem
[144] qq          16d22.9+c37.19   ; simplefor: simplefor, simplefor
[148] qq          11d22.9+c33.19   ; stepelem: stepelem, forelem
[152] qq          14d22.9+c33.19   ; whilelem: whileelem, forelem
[156] qq          13d22.9+c35.19   ; while: while, set warning
[160] qq          17d22.9+c35.19   ; endassign: endassign, set warning
[164] qq          20d22.9+c34.19   ; := : := . assign
[168] qq          12d22.9+c34.19   ; first:= : first:= . assign

```

d2:

```

d d5=e69-c30+4                      ; used at a12-1
d d18=4d2+e69, d19=d2-1, d20=1d2, d21=40e25-d2-523 ;
d e18=c36, i=39i, e16=k-e70        ;
e
d i=2e21, e22=e16-e15-1            ;
qq e15.9-e24.9+e18.19+e22.36+e22.34+205e13.39f, ;
d e15=e16                          ;
s

```

[5.1.66]

[Pass 3, page 1]

```
b k=e15+e70, i=205e13, a30, b3, c70, d50 ;
d e17=i ;
d d4=1022 [max stack] ;
```

[Input byte values]

```
d d13=229[↑], d17=151[;], d18=165[(] ;
d d7=221 [trouble] ;
```

[Output byte values]

```
d d50=200 [output base], d5=28[beg func] ;
d d9=8d50[end else exp], d10=2d50[proc,], d11=6[literal] ;
d d12=5[lit integer], d14=0 [CARRET], d15=124[do] ;
d d16=1016[dummy identifier] ;
```

[Stack representations]

```
d d3=19[beg block], d6=32[func], d8=10[else exp] ;
d d19=21[beg proc]
```

```
b: qq 1.3-1.39 ; mask. c10.
[1b] qq t 256 ; 1.0 floating, true. 1c60, c36, c40
[2b] qq 3 t 320 ; 10 floating. c38, c39, c41, c56
d i=i-e69-e69 ; CDC
[1b] qq ; true
[2b] qq 256.39 ; 256
d i=i+e69+e69-2 ;
m
[3b] qq 10.39 ; c56, in CDC: 4c39
[4b] qq 57 ; 57.c39, in CDC: 1c36, c38, 7c39, c51, 1a16
[5b] qq 1020.9+409.19+614.29+410.39 ; 0.1 floating. c43, in CDC: c38, 7c39, c40
d i=i-e69 ; CDC:
[5b] qq 1.39 ; 1
d i=i+e69-1 ;
[6b] hs c7 ; instruction, c7.
[7b] qq 1023.9+512.19 ; -1.0 floating, false, 1c60.
d i=i-e69 ; CDC
[7b] qq 1.39 ; false
[8b] qq 1023.39 ; 1023, 4e9
d i=i+e69+e69-2 ;
b1: qq ; decl.c13, c15, c16, c17, c20
[1b1]qq ; N. c60, c36, c39, c40, c48, a19, a9, c67, c59
; c56, in CDC: c36, 1c39., 1a16
[2b1]qq ; factor. c36, c38, c48 in CDC: c36, 2c39...
; 1c52, 1c53, c41, c48, a16
[3b1]qq ; x. c39, in CDC: c39, c40, 1c48
d d40=1 ; If redefined, d40=0: output of
s ; state, KA, KB=10, or stack = 00
```



```

c2: pi    0      V   -5   LQB   ; AFTER TROUBLE: if first after trouble then
      pi    0      t   -9      ;           first after trouble:= false else
      hv    c3      ;           introuble:= false; go to NEXT 1;
c66:hs    e3      ,    ps    i    ; OUT: output(par M);
c1: pa    [operand] DV      NQA   ; NEXT: operand:= 0;
      pa    c1      ,    hv    c2   ;   ifintrouble then go to AFTER TROUBLE;
c3: pmn   (e1)      X    1      ; NEXT 1: byte:= input;
      hs    e2      LA      ;   if byte < 512 then go to NOT OPERAND;
      hv    c5      NT      ;   if -, introuble then
a6: hs    e3[or a5] NQA   ;   begin if stack identifier then
      [if stack identifier then a5 else e3] ; stck id else output(byte) end;
      pa    c1      t    1      NQA   ;   if -, introuble then operand := 1;
c6: pmn   (e1)      X    1      ;   AFTER OPERAND: byte:= input;
      hs    e2      LA      ;   if byte ≥ 512 then alarm(51);
c5: ga    a1      V      NT      ; NOT OPERAND: R:= table[byte];
      hs    c7      ,    qq    s+e51 ;   marks:= marksof(table[byte]);
      hv    i+5      LKB      ;   skip output for KB = 1
      hs    e9      LKA      ;   Special state output: delimiter
      sy    27      LKA      ;           comma
      sy    (c1)     LKA      ;           operand
      sy    (i+3)    LKA      ;           state
d i=i-d40-d40-d40-d40-d40 ; remove special output unless d40=0
a1: pmn   [byte]    Xt    d2      ;   if marks > 1 then go to SPECIAL;
      hv    c9      LA      ;   OA:= bit(state, R); if marks = 0 then
a2: ck    27 [state] IOA   ;   begin byte:= byte - 1;
      hv    a3      LB      ;   R:= table[byte]; marks:= marksof(table[byte]);
      pmn   (a1)      X    -1     ;   OB:= bit(state, R);
      ck    (a2)      IOB   ;   if marks = 0 then
      hv    a4      LB      ;   begin byte:= byte - 1;
      pmn   (a1)      X    -1     ;   R:= table[byte];
      ck    (a2)      ;   if OB then byte := byte - 4
      qq    (a1)      t    -4     LOB  ;   end;
a4: qq    (a1)      t    -2     LOA   ;   if OA then byte:= byte - 2
a3: pmn   (a1)      XV    -1     LO   ;   end;
      pmn   (a1)      X      ;   if bit(state, R) then byte:= byte - 1;
      ck    (c1)      V      NC      ;   R:= table[byte]; marks:= marksof(table[byte]);
      hv    c8      ;   if marks > 0 then go to SEARCH;
      hv    c10     LQA      ;   if -, bit(operand, allowed operand part(R))
      pmn   (a1)      XV    LO      ;   ^ -, introuble then alarm (52);
      hs    c7      ,    qq    s+e52 ;
c10:mb    b      ,    ga    a2      ; NORMALACTION: state:= newstatepart(R);
c9: hv    c35      LC      ; SPECIAL: if marks = 3 then go to INITIALIZE NUMBER
      gt    a      ,    cl    -9     ;   parM:= outpart(R); parR:= stackpart(R);
a: ck    -11, hv [switching part]; go to instruction[switchingpart(R)];
a5: ck    -10      ,    gt    p1      ;   procedure stckid;
      pa    a6      t    e3      ;   begin special stack[ds+1]:= byte;
      ck    10      ;   stack identifier:= false;
      hs    e3      ;   output(byte),
      arn   d19      D      ;   par R:= beg proc;
      pm    b1      ,    hv    c18   ;   par M:= decl; ENTOUT end;

```

```

c63:pi    0      t    -13      ; START PASS 3: introuble:= first after trouble:=
      pm    a20    V          ;      stackidentifier:= false
a20:hv    c51    ,    hv    c51 ;      store[0]:= jump to error 2;
      gm    0      MA      ;      for i:= upper stack limit step -1 until 0 do
a21:grn   40e25  t    -1    M    ;      store with marks(stack[i], 0, 0);
      bs    (a21)  t    d1     ;      ds:= 0; state:= 27;
      hv    a21    ;      go to NEXT;
      pp    d1     ,    hh    c66 ;
c62:pm    (e1)    X    1      ; END PASS 3: output(input); comment final value
      hs    e2     LA      ;      of short from pass 2;
      ps    e92-1 ,    hv    e3 ;      transfer from drum (endpass track)
      ;      to:(place for endpass);
      ;      go to PLACE FOR ENDPASS;
c7:  arn    s      ,    ps    c66 ; ALARM: procedure alarm(n); value n; integer n;
      hv    c6     LQA     ;      begin if introuble then go to AFTER OPERAND;
      sr    6b     ,    ac    a10 ;      errormessage(n);
      hs    e5     ;      byteword:= byteword - 1;
a10:qq    d7     ,    qq     ;      byte:= trouble;
      pa    c1     ,    grn   a10 ;      introuble:= first after trouble:= true;
      pa    a10    t    d7     ;      operand:= 0; go to NOT OPERAND
      qq    (e1)   t    -1     ;      end;
      pi    12     t    -13    ;      comment errormessages: 38: stack, 51: -delimi-
      arn    a10   ,    hv    c5 ;      ter, 52: operand, 53: delimiter, 54: -operand,
      ;      55: number, 56: termination;
c8:  sy     64     NKC      ; Special output, stack: CARRET
      arn    (a1)   D      NKC ;
      hs    e9     NKC      ;      byte
      sy     27     NKC      ;      comma
      sy    (c1)    NKC      ;      operand
      pmn    (a1)   X      ; reset A and M
d i=i-d40-d40-d40-d40-d40-d40 ; remove special output
      ga    a11    V      LA   ; SEARCH: if marks = 1 then alarm(53);
      hs    c7     ,    qq    s+e53 ;      comment delimiter; base:= part 1(R);
      pm    (c1)   DX     ;      if marks = 3 then go to a12;
      hv    a12    LC     ;      if operand = 0 then alarm(54);
      pm    d8     DVX    NZ   ;      comment - operand;
a14:hs    c7     ,    qq    s+e54 ;      if stack[ds]  $\neq$  elseex then go to a13;
      nc    (p)    ,    hv    a13 ;
      sy    (p)    NKC      ; Special output: top of stack
d i = i - d40 ; remove special output unless d40 = 0
      pm    d9     DV      ;      R:= end else expr;
c55:cl    -9     X      ; c55:
      hs    e3     X      ;      output(R);
c54:pp    p-1    ; c54: ds:= ds - 1;
a13:sy    (p)    NKC      ; Special output: top of stack
d i = i - d40 ; remove special output unless d40 = 0
      pmn    (a1)   X      ; a13: if -,bit(stack[ds],
      ;      allowed stackpart(table[byte])
      ck    (p)    ,    is    (p) ;      then alarm(56); comment terminator;
a11:pmn    s[base] XV    LO   ;      R:= table[stack[ds] + base];
      hs    c7     ,    qq    s+e56 ;      marks:= marks of table[stack[ds] + base];
      hv    c10    NC     ;      if marks = 0 then go to NORMAL ACTION;
      hv    c55    LA     ;      if marks = 1 then begin R:= part 4(R);
      hv    c54    ;      go to c55 end; go to c54;

```

```

a12:hv    a13          LZ      ; a12: if operand = 0 then go to a13;
      nc    1          ;      if operand  $\neq$  1 then
      hs    c7          ,    qq  s+e52 ;      alarm(52);
      pm    d10        DX      ;
      hs    e3          ;      output(proc);
      hv    a13        ;      go to a13;
c11:pa    a6          t    a5      ; SET BLOCK PROC: stack identifier:= true;
c12:ca    (p)          ; SET BLOCK: if stack[ds] = parR then
      pa    p          t    d3      ;      stack[ds]:= ,begblock,;
c13:gm    b1          ,    hv    c1  ; SET DECL: decl:= parM; go to NEXT;
c14:pa    a6          t    a5      ; ADD DECL PROC: stackidentifier:= true;
c15:xr          ,    sc    b1      ; ADD DECL: decl:= decl + parM; go to NEXT;
      hv    c1          ;
c17:pm    b1          ; DECL ENT: parM:= decl;
c18:pp    p1          ,    bs    p-d4 ; ENT OUT: ds:= ds + 1;
      hs    c7          ,    qq    e38 ;      if ds > maxds then alarm(38); comment stack;
c19:ga    p          VX      ; CH OUT: stack[ds] := parR; go to OUT;
c20:pm    b1          X      ; DECL: parM:= decl; go to OUT;
      hv    e3          ;
c21:pp    p1          ,    bs    p-d4 ; ENT: ds:= ds + 1; if ds > maxds then
      hs    c7          ,    qq    e38 ;      alarm(38); comment stack;
c70:ga    p          ,    hv    c1  ; CH: stack[ds]:= parR; go to NEXT;
c65:qq    (e1)        t    -1      ; ENDPROCTR: byteaddress:= byteaddress - 1;
c22:arn    p          ,    tk    10  ; PROC END: output(special stack[ds]);
      hs    e3          ;
c61:qq    (2e4)        t    1      ; BLOCK COUNT: information 1:= information 1 + 1;
c23:pp    p-1          ; AN OUT: ds:= ds - 1; go to OUT;
c4: hv    e3          X      ;
c24:pp    p-1          ,    hv    c1  ; AN: ds:= ds - 1; go to NEXT;
c64:hs    e3          X      ; DO: output(par M);
      arn    d15        D      ;      parM:= ,do,;
      hv    c19        X      ;      go to CH OUT;
c26:bs    (c1)        ; LEFT PARENT: if operand > 0 then
      pmn    d5          D      ;      begin parR:= ,func,; parM,begcall, end;
      arn    d6          D      ;      go to ENT OUT;
      hv    c18        ;
c68:hs    e3          X      ; RIGHT CALL:
      arn    7d50        DX      ;
c25:pp    p-1          ,    ga    c1  ; RIGHT: ds:= ds - 1; operand:= parR;
      hsn    e3          X      ;      output(parM); go to AFTER OPERAND;
      hv    c6          ;

```

```

c69:xr      ,      ar  b1      ; BOUNDS: par M:= par M + decl;
      hv  c25      X      ;      go to RIGHT;
c27:bs  (c1)      ; PLUS: if operand = 0 then parM:= parR;
      pm  18d50    D      ;      go to OUT;
      hv  e3      X      ;
c28:bs  (c1)      ; MINUS: if operand > 0 then go to OUT;
      hv  e3      X      ;      sign:= parR; go to NEGATE;
      ga  a9      ,      hv  c36 ;
c29:bs  (c1)      ; BINARY: if operand = 0 v state > 7 v introuble
      bs  (a2)    t      7      NQA ;      then alarm(53);
      hs  c7      ,      qq  s+e53 ;      comment delimiter;
      ga  a2      X      ;      state:= parR; go to OUT;
      hv  e3      ;
c30:pi  8      t      -9      ; AN TROUBLE: introuble := true; ds:= ds - 1;
      pp  p-1    ,      hv  c6      ;      go to AFTER OPERAND;
c31:nsn (a6)      ,      nc  sa5      ; DECL PROC TR: if stack identifier
      hv  c1      ;      then begin output(trouble);
      hs  e3      X      ;      byte:= par M; stckid end;
      arn b3      ,      hv  a5      ;      goto NEXT;
b3:  qq  dl16      ;      dummy identifier
c32:qq  (e4)    t      1      ; CR: CRcounter:= CRcounter + 1;
      hs  e3      X      ;      output(parM);
      bs  (c1)      NQA      ;      go to if operand = 0 then NEXT 1
      hv  c6      ;      else AFTER OPERAND;
      hv  c1      ;
c35:bs  (c1)      ; INITIALIZE NUMBERS: if operand ≠ 0 then alarm(51);
      hs  c7      ,      qq  s+e51 ;      comment - delimiter;
      qq  (e1)    Vt  -1      NQA ;      if introuble then go to AFTER OPERAND;
      hv  c6      ;      byteaddress:= byteaddress - 1;
      pa  a9      ,      hv  c36 ;      sign is negative:= false; go to NEGATE;
c57:qq  (e4)    t      1      ; CR 1a: CRcounter:= CRcounter + 1;
      pmn dl14    DX      ;      output(,CARRET,);
      ps  i      ,      hv  e3      ;
c37:pmn (e1)    X      1      ; NEXT OF NUMBER: s:= input(byte);
      hs  e2      LA      ;      s:= if 56 < s ^ s < 67 then 57
      ga  r1      ;      else if 66 < s ^ s < 78 then s - 9
      ps      ;      else 56;
      bs  s445    t      501      ;      go to instruction[numbers +
      ps  57      ,      hh  r3      ;      bits(numberstate, numberstate + 4,
      bs  s434    t      500      ;      numberstatetable[s]);
      ps  s-9     ,      hh  r1      ;
      ps  56      ,      arn sd2     ;
a15:ck [numberstate], tk -5      ;
      ga  r1      ;
      hv [switchingpart]t c38      ;
s      ; Stop for choice
s      ;      between GIER and CDC
s      ;
s      ;

```

```

c60:ga    r1      X      ; LOGIC VALUE: N:= store[parR];
      pm      ,      gm 1b1 ; if -, introuble ^ operand ≠ 0 then alarm(1);
      bs (c1)      NQA ; comment -delimiter; operand:= 3;
      hs c7      ,      qq s+e51 ; kind:= parM;
      pa c1      t      3      ; go to OUTPUT VALUE;
      hv c67      ;
c33:pa    a7      X      3      ; LITERAL: for i:= 1 step 1 until 4 do
a8: pm    (e1)    X      1      ; begin input(byte);
      hs e2      LA      ; if -, introuble then output(byte)
      hs e3      NQA      ; end;
a7: bt      Xt    -1      ;
      hv a8      ;
      hs e3      NQA      ;
      pm (c1)    DX    3      NQA ; if -, introuble then R:= operand:= operand + 3;
      ca 3      ,      hv c6    ; if R = 3 then go to AFTER OPERAND;
      qq (e1)    t      1      NQA ; if -, introuble then byteaddress:=byteaddress+1;
      hs c7      ,      qq s+e51 ; alarm(51);
c36:grn 1b1      ,      pm 1b    ; NEGATE: N:= 0; facor:= 1;
      gm 2b1      ,      pa a15    ; numberstate:= 0;
      pa c1      t      3      ; exp 10:= 0; operand:= 3;
      pa a16      ,      hv c37    ; go to NEXT OF NUMBER;
c38:arnf 2b1      ,      mkf 2b    ; 0 DIGIT 2: factor:= 10×factor;
      grf 2b1      ,      it 15    ; 1 numberstate:= 1; go to DIGIT;
c39:pa    a15      t      5      ; 2 DIGIT 1: numberstate:= 5;
      arn (e1)    ,      sr 4b      ; 3 DIGIT: N:= N×10 + (byte - 57);
      nkf 9      ,      grf 3b1    ; 4
      arnf 1b1    ,      mkf 2b    ; 5
      arf 3b1     ,      grf 1b1    ; 6
      hv c37      ; 7 BLIND: go to NEXT OF NUMBER;
c52:hv    c56      ; 8 DIGIT 3: go to DIGIT 3a;
c40:pm    1b      ,      gm 1b1    ; 9 TEN 1: N:= 1;
c41:pa    a19      t      2b      ;10 TEN 2: pos exp:= true;
      it 20      ;11 numberstate:= 20; go to NEXT OF NUMBER;
c42:pa    a15      Vt    10      ;12 POINT: numberstate:= 10; go to NEXT OF NUMBER;
c53:pa    a15      t      35      ;13 ERROR 1: numberstate:= 35;
      hv c37      ;14 go to NEXT OF NUMBER;
c43:pa    a19      t      5b      ;15 EXP MINUS: pos exp:= false;
c44:pa    a15      t      25      ;16 EXP PLUS: numberstate:= 25;
      hv c37      ;17 go to NEXT OF NUMBER;
c45:hv    c57      ;18 CR 1: go to CR 1a;
c50:hv    c58      ;19 OUT 1: go to OUT 1a;
c51:hv    c59      ;20 ERROR 2: go to ERROR 2a;
      pa c48      V      d12      ;21 FINISH 1: kind:= ,lit integer,; go to COMMON FIN
c48:pa [kind] Dt    d11      ;22 FINISH 2: FINISH 3: kind:= ,lit real,;
      arnf 1b1    ,      dkf 2b1    ;23 COMMON FINISH: R:= N/factor;
a16:bt [exp 10] t    -1      ;24 for exp 10:= exp 10 - 1 while exp 10 > 0 do
a19:mkf[10 or. 1] , hv a16    ;25 R:= R×(if pos exp then 10 else 0.1);
      grf 1b1     ;26 N:= R;
a9: pm [sign] XD      ;27

```

hv	c49		LZ	; SIGN TREATMENT:
pm	(e1)	X		; if sign is negative ^ next symbol ≠ ,↑, then
ca	d13	X		; begin if byte = ↑ then output(sign)
ps	r1	,	hv e3	; else N:= -N
srnf	1b1	,	grf 1b1	; end;
c49:qq	(e1)	t	-1	; OUTPUT VALUE:
pm	(c48)	DX		; for i:= 0 step 10 until 30 do
c67:ck	-1	,	pa a17	; output(bit(i, i+9, N));
pm	1b1	X		; operand:= 3;
a18:cl	10	X		; output(kind);
ck	-10	,	hs e3	; byteaddress:= byteaddress - 1;
a17:btn		t	-120	; go to AFTER OPERAND;
ps	c66	,	hv c6	;
hv	a18	X		;
c58:qq	(e1)	t	-1	; OUT 1a: byteaddress:= byteaddress - 1;
pmn	(a9)	DX		; output(sign); go to NEXT;
ps	c66	,	hv e3	;
c59:grn	1b1	,	hs e5	; ERROR 2a: errormessage(10);
hv	c48	,	qq s+e55	; N:= 0; go to FINISH 1;
c56:pa	a15	t	30	; DIGIT 3a: numberstate:= 30;
arn	(e1)	,	it 35	; if exp 10 < 35 then
bs	(a16)	,	hv r5	; begin exp 10:= exp 10×10 + byte - 57;
sr	57	D		; go to NEXT OF NUMBER;
pm	(a16)	D		; end;
ml	3b	X		; exp 10:= 0; N:= 0;
ga	a16	,	hv c37	; if pos exp then go to ERROR 1;
pa	a16	,	grn 1b1	; go to NEXT OF NUMBER;
arn	a19	,	ca 2b	;
hv	c53			;
hv	c37			;
<u>s</u>				; End
<u>s</u>				; of
<u>s</u>				; GIER
<u>s</u>				; version

[Pass 3, page 10, stack words 1]

[New state.9+switching part.19+stack part.29+output part.39]

[stack: switching part, stack part, output]
[; 4 and ;7]

d d20=i-12

```
[12] qqf ; trouble: -, -, -
[13] qq 10d50.39; thenst: -, -, end thenst
[14] qq 11d50.39; goto: -, -, end goto
[15] qq 160.39; assign: -, -, end assign
[16] qq 15d50.39; do: -, -, end do
[17] qq 9d50.39; elsest: -, -, end elsest
[18] qq 27.9+ c4.19 + 120.39 ; begclean: OUT, -, ;
[19] qq 27.9+ c4.19 + 120.39 ; beg block: OUT, -, ;
[20] qq 27.9+ c4.19 + 120.39 ; beg body: OUT, -, ;
[21] qq 28.9+c22.19 + 104.39 ; beg proc: PROC END, -, end proc
[22] qq 28.9+c23.19 + 88.39 ; := switch: AN OUT, -, switch list
```

[end 1 and end 2]

d d21=i-12

```
[12] qqf ; trouble: -, -, -
[13] qq 10d50.39; thenst: -, -, end thenst
[14] qq 11d50.39; goto: -, -, end goto
[15] qq 160.39; assign: -, -, end assign
[16] qq 15d50.39; do: -, -, end do
[17] qq 9d50.39; elsest: -, -, end elsest
[18] qq 20.9+c23.19 + 96.39 ; beg clean: AN OUT, -, end clean
[19] qq 20.9+c61.19 + 100.39 ; beg block: BLOCK COUNT, -, end block
[20] qq 10.9+c24.19 ; beg body: AN, -, -
```

[else 1 and else 2]

d d22=i-11

```
[11] qq 3.9+c19.19+ 10.29+ 6d50.39 ; thenex: CH OUT, else ex, else ex
[12] qq 34.9+c30.19 ; trouble: AN TROUBLE, -, -,
[13] qq 9.9+c19.19+ 17.29+ 132.39 ; thenst: CH OUT, else st, else st
[14] qq 11d50.39; goto: -, -, end goto
[15] qq 160.39; assign: -, -, end assign
```

[.7 and parameter delimiter 1]

d d23=i-22

```
[22] qq 2.9+ c4.19 + 16.39 ; := switch: OUT, -, switchparam
[23] qq 2.9+c19.19+ 37.29+ d50.39 ; [arr: CHOUT, [arr,, bound comma
[24] qq 2.9+ c4.19 + 1d50.39 ; [left: OUT, -, [,
[25] qq 2.9+ c4.19 + 1d50.39 ; [for: OUT, -, [,
[26] qq 2.9+ c4.19 + 1d50.39 ; [subscr: OUT, -, [,
[27] qq 2.9+ c4.19 + 1d50.39 ; [left or subs: OUT, -, [,
[28] qq 2.9+ c4.19 + 144.39 ; := for: OUT, -, simple for
[29] qq 2.9+c19.19+ 28.29+ 148.39 ; until: CHOUT, := for, step elem
[30] qq 2.9+c19.19+ 28.29+ 152.39 ; while: CHOUT, := for, while elem
[31] qq 2.9+ c4.19 + 20.39 ; (call: OUT, -, call param
[32] qq 2.9+ c4.19 + 20.39 ; (func: OUT, -, call param
```

[Pass 3, page 11,stack words 2]
[right bracket 1]

```

    d d24=i-23
[23] qq 13.9+c69.19      +    72.39 ; [arr: BOUNDS, NEW OPERAND, bound base
[24] qq 27.9+c25.19+    2.29+36d50.39 ; [left: RIGHT, NEW OPERAND, ]
[25] qq 22.9+c25.19+    2.29+36d50.39 ; [for: RIGHT, NEW OPERAND, ]
[26] qq  1.9+c25.19+    2.29+36d50.39 ; [subscr: RIGHT, NEW OPERAND, ]
[27] qq  4.9+c25.19+    2.29+36d50.39 ; [left or subs: RIGHT NEW OPERAND, ]

    [do 1]
    d d25=i-28
[28] qq 27.9+c64.19+ 16.29+39d50.39 ; := for: DO, do, simple for
[29] qq 27.9+c64.19+ 16.29+40d50.39 ; until: DO, do, step elem
[30] qq 27.9+c64.19+ 16.29+41d50.39 ; while: DO, do, while elem

    [right parenthesis 2]
    d d26=i-31
[31] qq 20.9+c68.19      +    92.39 ; (call: RIGHT CALL, NEW OPERAND, end call
[32] qq  1.9+c25.19+    3.29+    92.39 ; (func: RIGHT, NEW OPERAND, end call
[33] qq  1.9+c25.19+    3.29+37d50.39 ; (subex: RIGHT, NEW OPERAND, )

    [then 1]
    d d27=i-34
[34] qq  5.9+c19.19+ 11.29+ 5d50.39 ; ifex: CH OUT, thenex, thenex
[35] qq  8.9+c19.19+ 13.29+ 128.39 ; ifst: CH OUT, thenst, thenst

    [step 1]
    d d28=i-28
[28] qq  2.9+c19.19+ 36.29+13d50.39 ; := for: CH OUT, step, step

    [until 1]
    d d29=i-36
[36] qq  2.9+c19.19+ 29.29+14d50.39 ; step: CH OUT , until, until

    [while 1]
    d d30=i-28
[28] qq  2.9+c19.19+ 30.29+ 156.39 ; := for: CH OUT, while, while

    [:3]
    d d31=i-37
[37] qq  2.9+c19.19+ 23.29+ 112.39 ; [arr,: CH OUT, [arr: , boundop 1
```


[trouble 1]

d d32=i-10

```
[10] qqf ; else ex: -, -, -
[11] qqf ; then ex: -, -, -
[12] qq ; trouble: -, -, -
[13] qq 34.9+c18.19+ 12.29+ 136.39 ; then st: ENT OUT, trouble, trouble
[14] qq 34.9+c19.19+ 12.29+ 136.39 ; goto: CH OUT, trouble, trouble
[15] qq 34.9+c19.19+ 12.29+ 136.39 ; assign: CH OUT , trouble, trouble
[16] qq 34.9+c18.19+ 12.29+ 136.39 ; do: ENT OUT, trouble, trouble
[17] qq 34.9+c18.19+ 12.29+ 136.39 ; elset: ENT OUT, trouble, trouble
[18] qq 34.9+c18.19+ 12.29+ 136.39 ; beg clean: ENT OUT, trouble, trouble
[19] qq 34.9+c18.19+ 12.29+ 136.39 ; beg block: ENT OUT, trouble, trouble
[20] qq 34.9+c18.19+ 12.29+ 136.39 ; beg body: ENT OUT, trouble, trouble
[21] qq 32.9+ c4.19 + 136.39 ; beg proc: OUT, -, trouble
[22] qq 32.9+c23.19 + 136.39 ; := switch: AN OUT, -, trouble
[23] qq 32.9+c23.19 + 136.39 ; [arr: AN OUT, -, trouble
[24] qqf ; [left: -, -, -,
[25] qqf ; [for: -, -, -
[26] qqf ; [subscr: -, -, -,
[27] qqf ; [left or subs: -, -, -
[28] qqf ; := for: -, -, -
[29] qqf ; until: -, -, -
[30] qqf ; while: -, -, -
[31] qqf ; (call: -, -, -
[32] qqf ; (func: -, -, -
[33] qqf ; (subex: -, -, -
[34] qqf ; if ex: -, -, -
[35] qqf ; if st: -, -, -
[36] qqf ; step: -, -, -
[37] qq 32.9+c23.19 + 136.39 ; [arr,: AN OUT, -, trouble
```

[Pass 3 page 13, control table for numbers]

[Input byte values from 56 to 68 have their control words c-marked (i.e. f and comma marked) and call for a special logic for analyzing numbers. While this logic is operating the input byte values are converted so as to give the appropriate entry of the following table. This is an action table having the current numberstate as the other argument. The action is given in the table as the machine address of the code relative to DIGIT 2 = c38. The number states are:

Number state	Position	
0	4	Before number
5	9	Following digit before point
10	14	Following point
15	19	Following digit after point
20	24	Following ten
25	29	Following exponent sign
30	34	Following digit after ten
35	39	In erroneous number

When an error has been detected the remaining part of the number is skipped. The error message is given on the following terminator.]

[Entry to table:		direct converted]	
<u>d</u> d2=i-56		[GIER]	
[56]	qqf 19.4+21.9+20.14+22.19+20.24+20.29+22.34+20.39, ;		<terminator>
[57]	qqf 2.4+ 2.9 + 8.24+ 8.29+ 8.34+ 7.39, ; 0		<digit>
[58]	qqf 12.4+12.9+13.14+13.19+13.24+13.29+13.34+ 7.39, ; 1		.
[59]	qqf 9.4+10.9+13.14+10.19+13.24+13.29+13.34+ 7.39, ; 2		10
[60]	qqf, ; 3		
[61]	qqf, ; 4		
[62]	qqf, ; 5		
[63]	qqf 19.4+21.9+20.14+22.19+16.24+20.29+22.34+20.39, ; 6		+
[64]	qqf, ; 7		
[65]	qqf, ; 8		
[66]	qqf, ; 9		
[67]	qqf 19.4+21.9+20.14+22.19+15.24+20.29+22.34+20.39, ; .		-
[68]	qqf 18.4+18.9+18.14+18.19+18.24+18.29+18.34+18.39, ; -		CAR RET
<u>d</u> a24=e69+e69+e69, a24=a24+a24 ;			
<u>d</u> a24=a24+a24+e69 ; a24:= <u>if</u> CDC <u>then</u> 13 <u>else</u> 0			
<u>d</u> i=i-a24 ;			
		[CDC]	
[56]	qqf 22.4+31.9+25.14+26.19+25.24+25.29+27.34+25.39, ;		<terminator>
[57]	qqf 2.4+ 2.9+ 0.14+ 0.19+11.24+11.29+11.34+10.39, ; 0		<digit>
[58]	qqf 17.4+17.9+13.14+13.19+13.24+13.29+13.34+10.39, ; 1		.
[59]	qqf 15.4+16.9+13.14+16.19+13.24+13.29+13.34+10.39, ; 2		10
[60]	qqf, ; 3		
[61]	qqf, ; 4		
[62]	qqf, ; 5		
[63]	qqf 22.4+31.9+25.14+26.19+19.24+25.29+27.34+25.39, ; 6		+
[64]	qqf, ; 7		
[65]	qqf, ; 8		
[66]	qqf, ; 9		
[67]	qqf 22.4+31.9+25.14+26.19+18.24+25.29+27.34+25.39, ; .		-
[68]	qqf 21.4+21.9+21.14+21.19+21.24+21.29+21.34+21.39, ; 10		CARRET
<u>d</u> i=i+a24-13 ;			

[5.1.66]

[Pass 3 page 14, main control table]

[Each input byte gives access to a word in the following table. If this word is NOT COMMA-MARKED the table contains NORMAL action words arranged as follows:

	Delimiter	action	word	p
	-	-	-	p-1
			
	-	-	-	1
	Delimiter	meaning	word	q
	-	-	-	q-1
byte value:	-	-	-	1
	Delimiter	meaning	comment.	

The delimiter meaning words in their bit no. w contain the number of the action word appropriate to the state w, in binary form, thus:

					Multiplier
q=1, p=1	Delimiter	meaning	word	1:	1
q=2, p=2 or 3	-	-	-	2:	1
	-	-	-	1:	2
q=3, p=4 to 7	-	-	-	3:	1
	-	-	-	2:	4
	-	-	-	1:	2

The action word number corresponding to the various states is also given in the delimiter meaning comment. Delimiter meaning word q is f-marked.

Delimiter action words come in two formats: No marks, simple action words:

qq allowed operand.3+new state.9+switching part.19+stack part.29+output.39

When the program indicated in the switching part is entered we have

R: qq stack part.9+(output:512).10+new state.29+switching part.39

M: qq (output rem 512).9

Comma-mark, search in stack action words:

qq(f) base address.9+allowed stack delimiter bits

Not f-marked words perform TEST FOR ELSE EXPRESSION. f-marked words perform TEST FOR PROCEDURE CALL. The base address points to the table of stack words.

If the word indicated by the input byte value is COMMA-MARKED we have a SPECIAL ACTION:

Not-f marked words supply:

qq switching part.19+parameter.29+output.39,

f-marked words enter the number reading program, controlled by the control table for numbers.]

[5.1.66]

[Pass 3 page 15, main control table]

```
qq 7.3+ 6.9+ c4.19 +18d50.39 ; +2: OUT, -, +
qq 15.3+ 6.9+c27.19+16d50.29+16d50.39 ; +1: PLUS, pos, +
qqf 31.5+ 1.7 ; 1
qq 1.6 ; 2: 72
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq 7.3+ 6.9+ c4.19 +19d50.39 ; -2: OUT, -, -
qq 15.3+ 6.9+c28.19+17d50.29+19d50.39 ; -1: MINUS, neg, -
qqf 31.5+1.7 ; 1
qq 1.6 ; 2: 76
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq c32.19 + 0.39, ; CARRET: CR, -, CR
qq c33.19 + 4.39, ; short string: LITERAL, -, lit string
qq c33.19+ 512.29+ 4.39, ; long string: LITERAL, -, lit string
qq c33.19 + 4.39, ; layout: LITERAL, -, lit string
qq ; not used
qq ; not used
qq ; not used
qq 8.3+ 7.9+ c4.19 +38d50.39 ; -,1: OUT, -, -,
qqf 31.5+ ; 1: 85
[01111 10000 1 00000 00000 2 00000 00000 3 00000 00000]
qq 8.3+ 2.9+ c21.19+ 14.29 ; goto 1: ENT, goto, -
qqf 12.11+3.28+3.31 ; 1: 87
[00000 00011 1 00000 00000 2 00000 00110 3 11000 00000]
qq 8.3+28.9+ c21.19+ 20.29 ; begin 2: ENT, beg body, -
qq 8.3+28.9+ c18.19+ 18.29+ 116.39 ; begin 1: ENT OUT, beg clean, begin
qqf 12.11+3.28+5.34 ; 1
qq 3.31+5.35 ; 2: 91
[00000 00011 1 00000 00000 2 00000 00110 3 22121 20000]
qq 8.3+22.9+ c4.19 +12d50.39 ; for 1: OUT, -, for
qqf 12.11+3.28+3.31 ; 1: 93
[00000 00011 1 00000 00000 2 00000 00110 3 11000 00000]
qq 8.3+ 2.9+ c70.19+ 34.29 ; if 4: CH, ifex, -
qq 8.3+ 2.9+ c18.19+ 34.29+ 3d50.39 ; if 3: ENT OUT, ifex, ifex
qq 8.3+ 2.9+ c70.19+ 35.29 ; if 2: CH, ifst, -
qq 8.3+ 2.9+ c18.19+ 35.29+ 4d50.39 ; if 1: ENT OUT, ifst, ifst
qqf 5.4+27.31 ; 1
qq 1.3 ; 4
qq 5.4+1.9 ; 2: 100
[00343 00002 1 00000 00000 2 00000 00110 3 11000 00000]
qq 8.3+25.9+ c12.19+ 18.29+ 52.39 ; own 1: SET BLOCK, beg clean, own
qqf 17.32 ; 1: 102
[00000 00000 1 00000 00000 2 00000 00010 3 00100 00000]
qq 8.3+15.9+ c13.19 + 69.39 ; integer 3: SET DECL, -, spec int
qq 8.3+17.9+ c13.19 + 53.39 ; integer 2: SET DECL, -, own integ
qq 8.3+19.9+ c12.19+ 18.29+ 45.39 ; integer 1: SET BLOCK, beg clean, decl integ
qqf 1.28+15.33 ; 1
qq 1.25+13.33 ; 2: 107
[00000 00000 1 00000 00000 2 00000 20010 3 33130 00000]
qq 8.3+15.9+ c13.19 + 70.39 ; real 3: SET DECL, -, spec real
qq 8.3+17.9+ c13.19 + 54.39 ; real 2: SET DECL, -, own real
qq 8.3+19.9+ c12.19+ 18.29+ 46.39 ; real 1: SET BLOCK, beg clean, decl real
qqf 1.28+15.33 ; 1
qq 1.25+13.33 ; 2: 112
[00000 00000 1 00000 00000 2 00000 20010 3 33130 00000]
```

[5.1.66]

[Pass 3 page 16, main control table]

```
qq 8.3+15.9+c13.19      + 71.39 ; Boolean 3: SET DECL, -, spec bool
qq 8.3+17.9+c13.19      + 55.39 ; Boolean 2: SET DECL, -, own bool
qq 8.3+19.9+c12.19+18.29+ 47.39 ; Boolean 1: SET BLOCK, beg clean, decl bool
qqf 1.28+15.33          ; 1
qq 1.25+13.33           ; 2: 117
[00000 00000 1 00000 00000 2 00000 20010 3 33130 00000]
qq 8.3+14.9+c15.19      + 4.39  ; procedure 4: ADD DECL, -, type proc
qq 8.3+14.9+c13.19      + 64.39 ; procedure 3: SET DECL, -, spec proc
qq 8.3+16.9+c14.19      + 4.39  ; procedure 2: ADD DECL PROC, -, type proc
qq 8.3+16.9+c11.19+18.29+ 40.39 ; procedure 1: SET BLOCK PROC, beg clean, proc
qqf 1.28+15.33          ; 1
qq 1.15                 ; 4
qq 1.19+13.33           ; 2: 124
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
qq 8.3+14.9+c15.19      + 8.39  ; array 4: ADD DECL. --arrah
qq 8.3+14.9+c13.19      + 62.39 ; array 3: SET DECL, --spec array
qq 8.3+24.9+c15.19      + 8.39  ; array 2: ADD DECL, -, array
qq 8.3+24.9+c12.19+18.29+ 38.39 ; array 1: SET BLOCK, beg clean, real array
qqf 1.28+15.33          ; 1
qq 1.15                 ; 4
qq 1.19+13.33           ; 2: 131
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
qq 8.3+14.9+c13.19      + 56.39 ; switch 2: SET DECL, -, spec switch
qq 8.3+23.9+c12.19+18.29+ 32.39 ; switch 1: SET BLOCK, beg clean, switch
qqf 1.28+1.32           ; 1
qq 13.33                ; 2: 135
[00000 00000 1 00000 00000 2 00000 00010 3 22120 00000]
qq 8.3+14.9+c13.19      + 76.39 ; string 1: SET DECL, -, string
qqf 13.33               ; 1: 137
[00000 00000 1 00000 00000 2 00000 00000 3 11010 00000]
qq 8.3+14.9+c13.19      + 72.39 ; label 1: SET DECL, -, spec label
qqf 13.33               ; 1: 139
[00000 00000 1 00000 00000 2 00000 00000 3 11010 00000]
qq 8.3+12.9+c13.19      + 80.39 ; value 1: SET DECL, -, value
qqf 9.33                ; 1: 141
[00000 00000 1 00000 00000 2 00000 00000 3 10010 00000]
qqf d20.9+1.11-1.22,    ; ;7: SEARCH STATEMENT
qq 4.3+31.9+c20.19      ; ;6: DECL, -, -
qq 8.3+28.9+ c1.19      ; ;5: NEXT, -, -
qq d20.9+1.11-1.22,    ; ;4: SEARCH IN EXPRESSION
qq 8.3+30.9+ c1.19      ; ;3: NEXT, -, -
qq 4.3+31.9+c13.19      + 84.39 ; ;2: SET DECL, -, formal
qq 4.3+28.9+c20.19      ; ;1: DECL, -, -
qqf 3.9+9.13+15.20+7.29-1.35 ; 1
qq 15.4+31.10+15.15+1.20+3.28+7.32+1.34 ; 4
qq 7.10+5.14+13.18+1.20+3.28+3.31+7.35 ; 2: 151
[04444 04477 1 70656 62131 2 70000 00770 3 77537 30000]
```

[5.1.66]

[Pass 3 page 17, main control table]

```
qq 8.3+28.9+c65.19      + 104.39 ; end 3: ENDPROCTR, -, end proc
qq d21.9+1.11-1.20,      ; end 2: SEARCH IN EXPRESSION
qqf d21.9+1.11-1.20,      ; end 1: SEARCH STATEMENT
qqf 3.9+1.20+3.28+15.35   ; 1
qq 15.4+3.7+5.35         ; 2: 156
[02222 02211 1 00000 00000 2 10000 00110 3 00131 30000]
qq d22.9+1.10-1.15,      ; else 2: SEARCH IN EXPRESSION
qqf d22.9+1.10-1.15,      ; else 1: SEARCH STATEMENT
qqf 1.8+1.20              ; 1
qq 15.4+7.7              ; 2: 160
[02222 22210 1 00000 00000 2 10000 00000 3 00000 00000]
qq 12.3+ 2.9+c26.19+33.29+34d50.39 ; (3: LEFT PARENT, subex, (
qq 4.3+ 2.9+c18.19+31.29+ 24.39 ; (2: ENT OUT, (call, beg call
qq 4.3+21.9+c13.19      + 84.39 ; (1: SET DECL, -, formal
qqf 31.5+3.7+1.16       ; 1
qq 1.0-1.9+3.28+3.31    ; 2: 165
[03333 33322 1 00000 01000 2 00000 00220 3 22000 00000]
qq d27.9+3.35,          ; then 1: SEARCH IN EXPRESSION
qqf 7.3+3.7             ; 1: 167
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
qq 4.3+ 9.9+ c4.19      + 48.39 ; :4: OUT, -, decl label
qq d31.9+ 1.37,         ; :3: SEARCH IN EXPRESSION
qq 4.3+ 8.9+ c4.19      + 48.39 ; :2: OUT, -, decl label
qq 4.3+27.9+ c4.19      + 48.39 ; :1: OUT, -, decl label
qqf 7.3+1.6+3.28+3.31   ; 1
qq      + 1.9           ; 4
qq 7.3+5.8              ; 2: 174
[03330 03024 1 00000 00000 2 00000 00110 3 11000 00000]
qq d28.9+1.28,          ; step 1: SEARCH IN EXPRESSION
qqf 7.3+1.6             ; 1: 176
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq d29.9+1.36,          ; until 1: SEARCH IN EXPRESSION
qqf 7.3+1.6             ; 1: 178
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq d30.9+1.28,          ; while: SEARCH IN EXPRESSION
qqf 7.3+1.6             ; 1: 180
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq d24.9+1.22-1.27,     ; ] 1: SEARCH IN EXPRESSION
qqf 7.3+1.6             ; 1: 182
[ 01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq 4.3+ 2.9+c18.19+25.29+35d50.39 ; [5: ENT OUT, [for, [
qq 4.3+ 2.9+c18.19+26.29+35d50.39 ; [4: ENT OUT, [subscr, [
qq 4.3+ 2.9+c18.19+27.29+35d50.39 ; [3: ENT OUT, [left or subs, [
qq 4.3+ 2.9+c18.19+24.29+35d50.39 ; [2: ENT OUT, [left, [
qq 4.3+ 2.9+c17.19+37.29          ; [1: DECL ENT, [arr, -
qqf 1.4+5.24                     ; 1
qq 7.3+7.7+1.22                  ; 4
qq 1.4+3.9+3.28+3.31             ; 2: 190
[04443 44422 1 00000 00000 2 00501 00220 3 22000 00000]
```

[5.1.66]

[Pass 3 page 18, main control table]

```
qq  d23.9+1.21-1.32, ; ,7: SEARCH IN EXPRESSION
qq  8.3+24.9+ c1.19 ; ,6: NEXT, -, -,
qq  4.3+14.9+ c1.19 ; ,5: NEXT, -, -,
qq  4.3+24.9+ c1.19 ; ,4: NEXT, -, -,
qq  4.3+21.9+ c1.19 ; ,3: NEXT, -, -,
qq  4.3+12.9+ c1.19 ; ,2: NEXT, -, -,
qq  4.3+17.9+ c1.19 ; ,1: NEXT, -, -,
qqf 7.3+3.7+3.15+21.21+1.33 ; 1
qq  7.3+3.7+7.15+1.24+1.33 ; 4
qq  7.3+3.7+3.13+1.21 ; 2: 200
[07770 07700 1 00265 50101 2 03004 00000 3 00050 00000]
qq  6.3+ 2.9+c18.19+28.29+ 140.39 ; := 4: ENT OUT, ...
qq  6.3+ 4.9+ c4.19 + 164.39 ; := 3: OUT, -, :=
qq  6.3+ 4.9+c18.19+15.29+ 168.39 ; := 2: ENT OUT , assign, first:=
qq  4.3+ 2.9+c17.19+22.29 ; := 1: DECL ENT, := switch, -
qqf 1.4+1.23 ; 1
qq  1.22 ; 4
qq  1.4+3.9+3.28+3.32 ; 2: 207
[00003 00022 1 00000 00000 2 00410 00220 3 22000 00000]
qq  d26.9+7.33, ; )2: SEARCH IN EXPRESSION
qq  1.1+18.9+c20.19 ; )1: DECL, -, -
qqf 1.21 ; 1
qq  7.3+3.7 ; 2: 211
[02220 02200 1 00000 00000 2 01000 00000 3 00000 00000]
qq  8.3+32.9+c22.19 + 104.39 ; trouble 7: PROC END, - , end proc
qq  8.3+35.9+c31.19 + 136.39 ; trouble 6: DECL PROC TR, -, trouble
qq  8.3+33.9+c20.19 ; trouble 5: DECL, -, -
qq  8.3+32.9+c20.19 ; trouble 4: DECL, -, -
qq  8.3+33.9+ c4.19 + 136.39 ; trouble 3: OUT, -, trouble
qq  8.3+32.9+ c4.19 + 136.39 ; trouble 2: OUT, -, trouble
qqf d32.9+1.9-1.37, ; trouble 1: SEARCH STATEMENT
qqf 31.5+31.10+11.15+1.18+7.22+19.31; 1
qq  5.12+15.17+5.21+1.24 ; 4
qq  9.13+5.18+5.25+11.31 ; 2: 221
[01111 11111 1 70525 56434 2 15124 20120 3 33000 00000]
qq ; not used
qq  d25.9+7.30, ; do 1: SEARCH IN EXPRESSION
qqf 7.3+3.7 ; 1: 224
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
```

[5.1.66]

[Pass 3 page 19, main control table]

```
qq c60.19+1b.29+ 7.39, ; true: Logic value, -, true
qq c60.19+7b.29+ 7.39, ; false: Logic value, -, -
qq c29.19+ 6.29+20d50.39, ; × : BINARY, -, ×
qq c29.19+ 6.29+21d50.39, ; / : BINARY, -, /
qq c29.19+ 6.29+23d50.39, ; † : BINARY, -, †
qq c29.19+ 6.29+22d50.39, ; ÷ : BINARY -, ÷
qq c29.19+ 1.29+24d50.39, ; < : BINARY, -, <
qq c29.19+ 1.29+25d50.39, ; ≤ : BINARY, -, ≤
qq c29.19+ 1.29+26d50.39, ; = : BINARY, -, =
qq c29.19+ 1.29+27d50.39, ; ≥ : BINARY, -, ≥
qq c29.19+ 1.29+28d50.39, ; > : BINARY, -, >
qq c29.19+ 1.29+29d50.39, ; ‡ : BINARY, -, ‡
qq c29.19+ 1.29+30d50.39, ; ^ : BINARY, -, ^
qq c29.19+ 1.29+31d50.39, ; v : BINARY, -, v
qq c29.19+ 1.29+33d50.39, ; ≡ : BINARY, -, ≡
qq c29.19+ 1.29+32d50.39, ; => : BINARY, -, =>
qq c62.19, ; endpass: ENDPASS, -, -
qq 4.3+21.9+ c1.19 ; param delim 2: NEXT, -, -
qq d23.9+3.32, ; param delim 1: SEARCH IN EXPRESSION
qqf 7.3+3.7 ; 1
qq 1.21 ; 2: 245
[01110 01100 1 00000 00000 2 02000 00000 3 00000 00000]

d d1=i ;

d e18=c63 ; start pass 3

d i=39i ;

d e16=k-e70 ; e16:= first free track

e

d i=e21+1, e22=e16-e15-1 ; e22:= number of tracks - 1
qqe15.9-e24.9+e18.19+e22.36+e22.34+205e13.39; pass 3 word to GP
d e15=e16 ;

s
```


[Synopsis

Pass 2 recognises byte strings representing identifiers and substitutes a unique byte for each such string. This is done regardless of block structure so that the same identifier will be represented by the same byte throughout the text.

The values of the bytes will be in the range $1023 > \text{<byte>} > 511$.

Standard identifiers are assigned fixed values in tight sequence starting from 1022. Other identifiers follow in tight sequence from last standard identifier.

Tables.

Pass 2 uses two tables: letter table[1:56] and table [first free after pass 2: top of store];

Table holds, starting at top of store, one word (referred to as short word) for each distinct identifier. All words representing identifiers with the same first letter form a chain whose links are the address parts. The starting point of each chain is given in the address part of the corresponding word in the letter table. A letter which does not start a chain is represented in the letter table by 0. The last word in a chain has bit 41 = 1 all others have bit 41 = 0;

The rest of the identifier is assembled as an integer in a base 67 number system.

If this integer $< 2^{30}$ it will be stored in the remaining 30 bits of the short word and bit 40 will be 1.

Otherwise the last 20 bits of the integer will be stored in bit 20 to 39 of short word and bit 40 will be 0. The rest of the integer is stored in one or more words (long words) starting at the first free word at the bottom of table and the address - 1 of the first of these words will have bit 0 = 1, the others, if any, will have bit 0 = 1.

Initialization

The entries in the tables pertaining to standard identifiers are formed during loading of the translator and are stored on drum starting on track e88.

track e88 word 0 holds:

qq address of last long word .9 +
number of tracks for long words - 1.19 +
number of tracks for short words .29 +
core address of first free short word .39;

words 1 to 14 hold the letter table in packed form with the entry for a in bits 0 to 9 of word 1, for b in bit 10 to 19 of word 1 and so on until entry for Ø in word 14 bit 30 to 39.

Word 14 has bit 40 = 1 the rest bit 40 = 0;

word 15 and onwards using as many tracks as necessary hold the long words for standard identifiers.

The following tracks hold the corresponding short words starting with the words to be loaded in 983 to 1022, 943 to 982 etc.

Pass 2 starts by unpacking word 0 to 14 of track e88. The value of e88 is given in 8e4h: ar , pp e88 and Pass two will leave the number of the last track for short words in 8e4h: for use by pass 6 which takes in the following tracks which holds the standard declaration table.

Table look up

Each identifier encountered by pass 2 is added to the table but the linking is not performed.

Pass 2 now compares each entry in the corresponding chain with the new identifier. If the identifier is found in the table the new identifier is removed from the table otherwise the necessary linking is performed, including clear of bit 41 of the hitherto last short word in the chain. Finally the address of the short word corresponding to the identifier is put out.

Program:]

b k=e15+e70, i= 205e13, a30, b20, d10;

d e17=i

[d1 = letter table[0|, defined at end

d2= chain list[0|, defined at end

d3 not used]

d d4=77 ; output value CR

d d5=78 ; - - short string

d d6=79 ; - - long string

d d7=80 ; - - layout

d d8=241 ; - - endpass

```

a1: pmn d2      X   1   IPC
a2h: pp  0      , pp  p1
      gm d1      t   1   M
      ga (r-1)  , bs  p508
      ck 10      , hh  a2
      hv a1      NPC
; initialize letter table: k:= 0;
;   for i:= 1, i+1 while PC = 0 do
;       begin R:= chain start list[i];
;       split(R, 40, 41, PC);
;       for p:= 1 step 1 until 4 do
;           begin k:= k + 1;
;           pack(letter table[k], 0, 9, part(p, R),
;               10, 39, 0, 40, 41, 0)
;       end unpack one word from list;
;   end;

a3: vk  0      , ps  a3
a4: pmn(e1)    X   1
      hs e2      LA
      ga b1
b1: bs  -1      V   56  NT
      hh (b1)   X   a16
      hv e3
b2: ppn -1      IZC
; start byte processing: wait for track;
; next: byte:= first char:= input;
;   if byte > 56 then not identifier:
;       begin if byte > 511 then
;           begin R:= 0; M:= neg; go to act[byte-1018] en
;           output(byte); go to next
;       end;
; start identifier: p:= i long; R:= 0;
;   short:= t; ZC:= 3;
; nextchar:
;   RM:= R; byte:= input; finis:= t;
;   if byte > 66 then CR or end identifier:
;       begin if byte = input CR then go to CR;
;       R:= RM; go to end identifier
;   end;
;   RM:= RM×67 + byte; finis:= f;
;   if RM < 2↑39 then word not full:
;       begin R:= RM; go to next char end;
;   if -, short then store long word:
;       begin table[p+1]:= remain(RM, 2↑39);
;       p:= p + 1; R:= RM:2↑39; go to next char
;   end;
; store first long: a6: short:= f; ZC:= 1;
;   pack (table[i short], 0, 9, 0, 10, 19, p,
;       20, 39, remain(RM, 2↑20), 40, 41, ZC);
;   R:= RM:2↑20;
;   if -, finis then go to next char;
; end identifier:
;   if short then try store short:
;       begin RM:= R; if part 1(R) ≠ 0 then
;           go to store first long; M:= mask short;
;           pack(table[i short], 0, 39, RM, 40, 41, ZC)
;       end
;   else store last of long:
;       begin table[p+1]:= R + 2↑39;
;       p:= p + 1; M:= mask long
;   end;

a5: pm (e1)    X   1
      hs e2      LA
      ga b3
b3: bs  -1      V   66  NT
      ca 1020    , hv  a15
      hv a7      X
      ck 10      , ml  b10
      hv a5      X   LZ
      gm p1      V   NZA
a6: cl  19      XV   IZA
      pp p1      , hv  a5
      ck -19
      gr (b6)    X   MZC
      it p       , pt (b6)
      hv a5      LA
;
a7: nc  0      XV   LZA
      ar 512     DV
      hvn a6
      gr p1      V   NZA
      gm (b6)    MZC
      pm b11     V   LZA
      pm b12     , pp  p1

```

```

b4h:bs p-511 t -1      ; test storage: if p-511 > i short test then
hv a14                ; goto table overflow;

qq (e1) X -1          ; start look up: reset input to same byte;
pm (b1) XV d1 IPC     ; R:= letter table[first char];
a8: arn(b5) V IPC     ; split(R, 40, 41, PC); goto if R = 0 then
hv a12 LZ            ; first identifier in chain else look up;
; after long failed: a8:
a9: ga b5 XV NPB      ; R:= table[chain]; split(R, 40, 41, PC);
hv a13              ; look up: a9: if PB = 0 then chain := part 1(R)
b5: pm -1 X IPC       ; else go to new identifier;
b6: cm 512          ; R:= table[chain]; split(R, 40, 41, PC);
hv a9              ; if mask(M, R) ≠ mask(M, table[ishort]) v
qq V NPA            ; -, (short ≡ PA = 1) then go to look up;
hv a9 NZA           ;
hv a11 LPA          ; agree on short: if short then go to found;
hv a9 LZA           ;
it (b2) , pa b7      ; test long: i long 1:= i long;
ck 10 , ga b8        ; ilong 2:= part 2(R);
a10:                ; test next long: i long 1:= i long 1 + 1;
b7: arn -1 t 1 ITA    ; R:= table[i long 1]; TA:= R ≥ 239;
b8: sr -1 t 1        ; if R ≠ table[i long 2] then
hv a8 NZ            ; go to after long failed;
hv a10 NTA          ; if -, TA then goto test next long;
a11:pm (b5) DX       ; found:
hv e3              ; output(chain); go to next;

; first identifier in chain:
a12:it (b1) , pa b5   ; chain:= first char + base letter table
a13:pm (b6) DX        ; new identifier: connect to chain
ga (b5) MPA          ; pack(table[chain] 0, 9, i short, 41, 41, 0);
gp b2 NZA           ; if -, short then i long:= p;
is (b6) t -1        ; outbyte:= i short; i short:= ishort - 1;
it s-512 , pt b4     ; i short test:= i short - 512;
qq (2e4) t 1        ; inf 1:= inf 1 + 1; if outbyte > 511 then
hv e3 LT           ; begin output(out byte); go to next end;
a14:hs e5           ; table overflow:
qq , qqn e37        ; alarm(⌊<too many identifiers⌋, 8+ 0);

a15:qq (e4) X 1      ; CR: CRcounter:= CR counter + 1
pm d4 D            ; if M = neg then
hv e3 X LT        ; begin output(out CR); go to next end
hs e3 X           ; output(outCR);
hv a5 X           ; go to next char;

; constants: comment
b10:qq 67.39,      ; 67
b11:qq 1.9 - 1.39 ; mask short
b12:qq 1.19 - 1.39 ; mask long;

```

```
[switch act:= endpass, free CR, sh str, lg str, layout;]
```

```
[-5]pm d8 DX ; endpass: R:= outend; go to end pass 2;
[-4]hv a17 , hv a15 ; free CR: go to CR;
[-3]pm d5-d7 DXV ; sh str: R:= outshstr - outlay; go to layout;
[-2]pm d6 DXV ; lg str: R:= outlgstr; go to outcopy
[-1]ar d7 D ; layout: R:= R + outlay;
a16:pp 4 , pp p-1 ; outcopy: p:= 4; rep copy: p:= p - 1;
hs e3 ; output(R);
pmn(e1) X 1 ; R:= input;
hs e2 LA ;
bs p , hh a16 ; if p > 0 then go to rep copy;
hv e3 ; output(R); go to next;
a17:
b13:nt -1 , it (b2) ; endpass 2: inf 2:= i long - initial i long;
pa 3e4 , hs e3 ; output(R);
pmn(b6) DX ;
d1: ps e92-1 , hv e3 ; output(i short); go to endpass track

; commend the following is overwritten by
; initialize letter table;
a18:qq , ud 4e4 ; start pass 2: p:= standard table track;
vk p , lk d2 ; call track(p, chain start list[0];
vk p , arn d2 ; unpack inf: R:= chain start list[0];
ga b2 , ga b13 ; i long:= initial i long:= part 1(R);
gt b14 , tk 20 ; rest long tracks:= part 2(R);
ga b16 , gt b4 ; short tracks:= part 3(R);
tk 10 , ac b6 ; i short test:= part 4(R);
; i short:= 512 + part 4(R);
b14:gp b15 , pp -1 ; trackno:= p; i:= 0;
a19:hv a20 , pp p-1 ; for p:= rest long track step - 1 until 0 do
b15:vk -1 t 1 ; begin trackno:= trackno + 1; i:= i + 40;
lk d2 t 40 ; call track(trackno, chainstartlist[i])
a20:bs p , hh a19 ; end;
; i:= top table
b16:pp -1 , hv a22 ; for p:= short tracks step -1 until 0 do
a21:pp p-1 , ud b15 ; begin trackno:= trackno + 1; i:= i - 40;
lk 40e25 t -40 ; call track(trackno, table[i])
a22:bs p , hv a21 ; end;
it (b15) , pt 4e4 ; standard table track:= trackno;
hv a1 ; go to initialize letter table;

d d2=d1+42 ; define chain start list[0];
d e91=d2 ; define chain start list[0] for use by TLA;
d e18=a18, i=39i, e16=k-e70; Information for password

e ; end pass 2

d i=e21, e22=e16-e15-1 ; Load password
qq e15.9-e24.9+e18.19+e22.36+e22.34+205e13.39;
d e15=e16 ; set first free track

s
```

[CONTENTS OF TAPE:

1. Specifications of indicator use and switches	2
2. Definitions of output values used outside tables	3
3. Definitions of buffer areas for long strings Reservations for 2 variables, compound table, input table lower case	3
4. Program	
4.1. Main block (a30, local labels)	4
4.2. Reservations for input table upper case	9
4.3. String block (a40, local labels)	10
4.4. Type in block. (a10, local labels)	14
5. Tables	
5.1. Formats	15
5.2. Programs for compound symbols not identifiable on first char. Define top pass 1. Define typebuf[0	16
5.3. Redefinition of normal actions	17
5.4. Compound table. Is read to D.	17
5.5. Input table lower case. Is read to d1:	18
5.6. Input table upper case. Is read to d2:	20
6. End of tape	21

Special requirements: Between input table lower case [0| and input table upper case [0| must be at least 256 words as the range of an input symbol is 0 to 255 (when parity check is given as a bit in the symbol).

Allowed input bytes are in the table flagged by an f mark. All input bytes outside tables are forbidden. Consequently no word outside tables (from input table lower case [0| to input table lower case [255| and from input table upper case [0| to input table upper case [255|) may be f-marked.

end contents...]

[1. SPECIFICATIONS OF INDICATOR USE AND SWITCHES]

[INDICATOR USE

NZA \equiv normal, initialized to false
LZB \equiv comm ok. In strings: LOB \equiv set case; initialized to false
LTA \equiv line print, initialized by initialize translator
NTB \equiv ignore poff - - -
LPA \equiv print initialized to true
LPB \equiv sum or print - - -
LQA \equiv lined
LQB \equiv comp
NRA \equiv first word (in strings)
NRB \equiv type input, initialized by initialize translator

SWITCHES, elements given as <identifier> = <slipname>

switch special action:=

LC = c6, UC = c7, line = c8, bar = c9, CR = c10, poff = c11, pon = c12,
endcode = c13, sumcode = c14, clearcode = c16;
These actions are performed independent of context (the actions themselves
are not necessarily context independent).

switch normal action :=

connect = c20, semicolon = c21, end = c22, begin = c24, right parent = c25,
colon = c26, equal = c27, minus = c28, error in normal = c29, space in
normal = c30, next = c2;
The slipnames are redefined to <slipname> - c just before loading of tables
so that the values in the tables will have a negative sign bit. This im-
plies that c - c2 < 512 (c = 1c30 which is the last normal action, and c2
is the first normal action).

switch compound action :=

rest 8 = c15, rest8 no = 1c15, rest7 = 2c15, rest7 no = 3c15, rest6 = 4c15,
rest 6 no = 5c15, rest5 = 6c15, rest5 no = 7c15, rest4 = 8c15, rest4 no =
9c15, rest3 = 10c15, rest3 no = 11c15, rest2 = 12c15, rest2 no = 13c15,
rest1 = 14c15, rest1 no = 15c15, treat normal = c4, black comp err = c17,
red long comp err = c18, red short comp err = c19, string or layout = c40,
lined b = c60, lined e = c61, lined f = c62, lined g = c63, lined i = c64,
lined s = c65, lined t = c66;

switch layout action:=

d = c41, zero = c42, n = c43, point = c44, ten = c45, space = c46, minus
= c47, plus = c48, lined plus = c49, terminator = c50, all others = c51
= a;
(a = c51 defined just before table loading).

The table representations are absolute addresses (except for normal ac-
tions where they are relative to c) and not switch position numbers]

b k=e15+e70, i=205e13, a, b60[variables], c80[labels], d50[address constants];

[2. DEFINITIONS OF OUTPUT VALUES USED OUTSIDE TABLES and other usage of d names]

d 4 = 0; base for output values for letters
[d5 is defined after 5.2 to top pass 1. d6 - d8 not used]
d 9 = 17; value of by for type in
d10 = 63; CR (in strings);
[d11 is defined just before tables, not output value]
d12 = 151; ;
d13 = 156; end
d14 = 1019; endpass
d15 = 231; <
d16 = 200; ,
d17 = 14; _ in strings
d18 = 235; >
d19 = 10; } string end in strings
d20 = 1021; short string
d21 = 1022; long string
d22 = 1020; CR
d23 = 211;)
d24 = 91; begin
d25 = 245; fat ,
d26 = 1023; layout
d27 = 174; :
d28 = 233; =
d29 = 76; -
d30 = 207; :=
d31 = 240; =>
d32 = 85; -,
[d33 is defined just before tables, d34 - d36 are not used, d37 - d39
are used internally in typin, d40, d41 internally in string]

[3. DEFINITIONS OF BUFFER AREAS, RESERVATIONS FOR TABLES]

d42 = 1e13 ; Alternating buffer areas for long strings,
d43 = 42e13 ; same as input buffers in general pass

b: qq ; Compound symbol.
qq ; Out CR; bit 41 = 1 \equiv clearcode has been read

d = i ; Compound table[0]
i = 38i ; reserve compound table[0:37], loaded after program.

d1 = i ; Input table lower case[0]
i = 65i ; reserve input table lower case[0:64],
; loaded after program

d3 = i - 1 ; input table lower case[64]
i = 3i ; reserve space for 3 secondary table words (_ | CR)

[4.1. MAIN BLOCK]

b a30

```

c1: pm  r      V      IQC      ; next 1: comp:= lined:= f; go to next;
      , ps i      ; comment ps i will make the first return, which
      ; does not match a call, goto next;

c2: lyn b1      V      LRB      ; next: if -, type input then char:= read a char
      hv c72      ; else go to get a typed char;
c73:      ; typechar: setadr0[M, char];
b1: pm  -1      X  d1      ; char:= char + case; R:= table[char];
      hv a3      X  NB      ; test allowed;
      hv a4      X  LPB      ; test sum or print;
a1: hv (b1)      LA      ; test special char;
a2: hr s1      LZA      ; after spec: if -, normal then return;
c3: hv a5      LQB      ; innormal: testcomp;
c4: mb 255      DV NT      ; treat normal: if bit(R, 0)  $\neq$  1 then
c5: ga r2      V      ; normal out: begin normal:= t; commok:= f
      hv e3      IZC      ; output(bits(R, 2, 9)); return end;
      hv -1      t c IZA      ; normal:= t; go to normal action[part 1(R)];
      ; procedure test allowed;
a3: nc 127      , ca 63      ; if bit(R, 41)  $\neq$  1 then
      hv c2      ; begin if part1(M)  $\neq$  127  $\wedge$  part1(M)  $\neq$  63
      hs c70      ; then alarm( $\{<$ character $\}$ , 0+1);
      hv c2      , qq se31 ; go to next end;

a4: ga b5      ; procedure test special char;
b11: ar -1      D      ; if sum or print then
      ga b11      ; begin sum 1:= sum 1 + part 1(M);
b12: qq -1      t 1 LO      ; if overflow then sum 2:= sum 2 + 1;
b5: sy -1      LPA      ; if print then outchar(part1(M));
      arn(b5)      D LZA      ; end;
      hv a1      X      ;

      ; procedure test special char;
      ; if bit(R, 40) = 1 then
      ; go to specialaction[char];

      ; procedure test comp;
a5: hv a6      NQA      ; if lined then
      gt r      , pm -1      ; begin comp:= lined:= f;
      gm b      X IQC      ; comp symbol:= table[part2(R)];
      gt r      , hv -1      ; go to compaction[part2(compsymbol)]
      ; end
      ; else if comp then
a6: gt r      , is -1      ; begin comp:= f;
      pm s1      X IQB      ; R:= table[part2(R)+1];
      ; if bit(40, R)  $\neq$  1 then R:= table[undefined bar]
      pm 37d      X NA      ; go to compaction[part2(R)]
      gt r      , hv -1      ; end;

```

```

c6: pt  b1      V    d1      ; LC: case:= baseLC; go to next;
c7: pt  b1      t    d2      ; UC: case:= baseUC; go to next;
    hv  c2                      ;

c8: qq                V    LQB  ; line:
    hv  c2                IQC  ;   if -,comp then comp:= lined:= t;
    hv  c2                LQA  ;   if lined then go to next;
    arn 1d3      , hv  a2  ;   R:= table[second line word]; go to after spec;

c9: arn 2d3      V    LQA  ; bar:
    hv  c2                IQB  ;   if -,lined then
    hv  a2                      ;   begin comp:= t; go to next end;
                                ;   R:= table[second bar word]; go to after spec;

c10:pm  1b      X          ; CR:
    pm (e4)    DV  1 NZ    ;   if outCR = 0 then go to next;
    hv  c2                      ;   line counter:= line counter + 1; output(out CR);
    hs  e3                IPC  ;   sum or print:= bit(outCR,41)=1; print:=f;
    hv  a8                LTA  ;   if linecount then go to count 10 lines;

a7: pm  d10     DXV  LZA  ; after count: if -, normal then set adr0(M,<stringCR>)
    hv  c2                NQA  ;   else if -, compound then go to next;
    pm  3d3      X          ;   R:= table[second CR word];
    hv  a2                      ;   go to after spec;

a8: bt  0        t  -55    ; count 10 lines: ten count:= ten count + 1
    sy  58      , hh  r1    ;   if tencount = 10 then
    hv  a7      , pa  r-2   ;   begin outchar(58); ten count:= 0;
    hs  e7      X          ;   test print with CR(line counter);
    arn b1      , ck  10    ;   if case = base(UC) then outchar(60);
    ca  d2      , sy  60    ;   outchar(0); sum or print:= print:= t;
    pm  d3      IPC        ;   end;
    sy  0      , hv  a7    ;   go to after count;

c11: hv  c2                NTB  ; poff: if ignore poff then go to next;
    hs  c70                      ;   alarm({<off>, 0+3});
    hv  r1      , qq  pe43  ;
    can s-i-2 , hv  c2      ;   if in poff then go to next;
    gi  b2                      ;   save (normal, commok, comp, lined);
a9: hsn c2                IZA  ; rep off: normal:= f; inpoff:= t; call(next);
    hv  a9                      ;   go to rep off;

c12: hv  c2                NTB  ; pon: if ignore poff then go to next;
    hs  c70                      ;   alarm({<on>, 0+3});
    hv  r1      , qq  pe44  ;
    ncn s-a9 , hv  c2      ;   if -,in poff then go to next;
b2: pi  -1        t  243    ;   restore(normal, commok, comp, lined);
    hr  c2                      ;   return to(next);

```

```

c13:hs  c70          ; endcode: alarm(⌊<pause⌋, 8+3);
          , qqn pe33 ;
a10:      , ud  8e4   ; pause:  by:= typewr by;
      lyn b1      , vy (8e4) ; char:= read a char; by:= normal by;
      ca 19 [t], hv  c71   ;   if char = <t> then go to typin;
      hv  c2          ;   go to next;

c14:arn b11      , tl  -10 ; sumcode:
      ar (b12)  D        ;   check char:=
      tl  -59      ;   remain(sum1 + sum2×2↑10, 127);
      dl  b6      , cl  -10 ;
      ga  b4      , lyn b1 ;   char:= read a char;
      pa  b11     , pa  b12 ;   sum 1:= sum 2 := 0;
b4: ca  1      , hv  c2   ;   if char = checkchar then go to next;
      hs  c70      ;   alarm(⌊<sum⌋, 0+3);
      hv  c2      , qq  pe57 ;   go to next;

          ; comment
b6: qq  127.39      ; constant for forming check character;

c15:hs  a12      NQA    ; rest8: if -, lined then call(compsterr);
      hsn c1      IZA    ; rest8no: normal:= f; call(next1);
      hs  a12      NQA    ; rest7:
      hsn c1      IZA    ; rest7no:
      hs  a12      NQA    ; rest6:
      hsn c1      IZA    ; rest6no:
      hs  a12      NQA    ; rest5:
      hsn c1      IZA    ; rest5no:
      hs  a12      NQA    ; rest4:
      hsn c1      IZA    ; rest4no:
      hs  a12      NQA    ; rest3:
      hsn c1      IZA    ; rest3no:
      hs  a12      NQA    ; rest2:
      hsn c1      IZA    ; rest2no:
      hs  a12      NQA    ; rest1:
      hsn c1      IZA    ; rest1no:
      hs  a12      NQA    ; rest0:
a11:pmn b      X      IQC ; after comp: R:= compsymbol;
b7: hv  c4          ;   comp:= lined:= f;
          ;   go to if no comperr then treat normal
          ;   else black comperr

c16:acn 1b      MB      ; clearcode: pack(out CR, 40, 41, 1)
      pa  b11     , pa  b12 ;   sum 1:= sum 2:= 0;
      hv  c2      IPB    ;   sum or print:= t; go to next;

c31:          ;
a12:pa  b7      t      c17 ; comp rest error: no comp err:= f;
      ck  -7          ;   if bit(R, 33 ) = 1 then
      hr  s1      LO    ;   return; comment letter without line;
      hr  c17      ;   return to(comp err);

c17:pa  b7      t      c4  ; comp err: no comp err:= t;
      hs  c70      ;   alarm(⌊<compound⌋, 0+1);
      hv  a11     , qq  se32 ;   go to after comp;

```

```

c18:hsn c1      IZA      ; long red comperr: normal:= f; call(next1);
      hv c18      LQB      ;   if comp then go to long red comp err;
c19:gr b        , hv c17  ; short red comp err: comp symbol := R; go to comp err;

c20:hs c70      NZB      ; comment: if -,commok then
      hv a15      , qq se35 ;       alarm(⌊<comment⌋,0+1);
a15:hsn c2      IZC      ; rep comm: normal:= f; commok:= t; call(next);
      nc d11[;] , hv a15    ;   if part 1(R) ≠ <;> then go to rep comm;
      hv c1       IZA      ;   normal:= t; go to next 1;

a16:pm r        IQC      ; after end comm: comp:= lined:= f;
c21:pmn d12     DX      IZB ; semicolon: commok:= t; normal:= t;
      hv e3       IZA      ;   output(outsemicolon); return;

a17:hs c1       ; test end: call next;
      nc 4d4[d]   LQA      ;   if part 1(R) ≠ <d> ∨ -, lined then
      hv a19      ;   go to test sem;

c22:pmn d13     DX      ; end: output(outend); commok:= f;
      hs e3       IZB      ;
b10:bt 2        t 1     ;   endlevel:= endlevel + 1
      hv a20      ;   if endlevel > 1 then go to postlude;
a18:hsn c1      IZA      ; rep end comm: normal:= f; call(next1);
                        ; test sem:
a19:ca d11[;] , hv a16   ;   if part1(R) = <;> then go to after end comm;
      nc 5d4[e]   LQA      ;   if part1(R) ≠ <e> ∨ -,lined then
      hv a18      ;   go to rep end com;
      hsn c1      ;   call next1;
      hv a19      NQA      ;   if -,lined then go to test sem;
      ca 14d4[n] , hv a17 ;   if part 1(R) = <n> then go to test end;
      nc 12d4[l] , hv a19 ;   if part 1(R) ≠ <l> then go to test sem;
      pm 14d      , gm b    ;   comp symbol:= table(<else>);
      hv 13c15    ;   go to rest2no;

a20:pa 1b      , ud 10e4 ; postlude: outCR:= 0;
      hs e5       ;   if pass inf then
      hv r1      , qqn(pe39) ;       alarm(⌊<end⌋, 12+3);
      arn b1     , ck 10    ;   if case = base UC then outchar(60);
      ca d2      , sy 60    ;
      pm d3      IPC      ; sum or print:= print:= t;
      hsn c2      IZA      ; rep postlude: normal:= f; call(next);
      nc d11[;] , hv r-1   ;   if part1(R) ≠ <;> then go to rep postlude;
      pmn d14     DX      ;   output(<endpass>);
      sy 58      , hs e3    ;   outchar(<LC>);
      vk (4e4)   , sk (b45) ;   to drum(stringtrack, start buf);
      it (b39)   , pt 7e4   ;   string no:= word count;
      bs (3e4)   , it 39    ;   inf 1:= word count +
      it (b39)   t 36      ;   (if inf 2 > 0 then 39 else 36);
      pa 2e4     , hv e92   ;   go to end pass;

```

```

c74:                                ; start pass 1:
c23:hs  c1      , qq  c2-1          ; prelude: call(next1); set return(next); test b:
a21:nc  2d4[b]    LQA                ;   if part1(R)  $\neq$  repr(<b>)  $\vee$  -,lined then
    hv  c23                        ;   go to prelude;
    hs  c1      LQA                ;   if lined then call(next1);
    nc  5d4[e], hv a21              ;   if part1(R)  $\neq$  repr(<e>) then go to test b;
    hs  c1      LQA                ;   if lined then call(next1);
    nc  7d4[g], hv a21              ;   if part1(R)  $\neq$  repr(<g>) then go to test b;
    hs  c1      LQA                ;   if lined then call(next1);
    nc  9d4[i], hv a21              ;   if part1(R)  $\neq$  repr(<i>) then go to test b;
    hs  c1      LQA                ;   if lined then call(next1);
    nc  14d4[n]   LQA                ;   if part1(R)  $\neq$  repr(<n>)  $\vee$  -, lined then
    hv  a21                        ;   go to test b;
    pm  d22      DX                  ;   out CR:= outCR + outvalue CR; print:= f;
    ac  1b      IPC                  ;   sum or print:= bit(OUTCR, 41) = 1;
    pm  r      IQC                  ;   comp:= lined:= f;
c24:qq  (b10)   t    -1              ; begin: endlevel:= endlevel - 1;
    pmn d24      DX  IZB              ;   commok:= normak:= t;
    hv  e3      IZA                  ;   output(outrepr(<begin>)); return;

c25:hsn c2      IZA                  ; right parent: normal := f; call(next);
    ck  -2      NQB                  ;   if -,comp  $\wedge$  bit(R, 38) = 1 then
    hv  a22      LO                   ;   go to maybe fat;
    ck  2      NQB                  ;
    pm  d23      DX                  ;   output(outrepr(<>));
    hs  e3      IZC                  ;   normal:= t; commok:= f;
    hv  c3      X                    ;   go to innormal;

a22:ck  -2                        ; maybe fat:
    hv  c25      LO                   ;   if bit(R, 36) = 1 then go to right parent;
a23:hs  c2                        ; in fat: call(next);
    ck  -2      NQB                  ;   if bit(R, 38) = 1  $\wedge$  -,comp then
    hv  a23      LO                   ;   go to in fat;
    ck  -3      NQB                  ;   if bit(R, 35)  $\neq$  1  $\vee$  comp then
    hv  a25      NO                   ;   go to fat error;
a24:hs  c2                        ; end fat: call(next);
    ck  -4      NQB                  ;   if bit(R, 36) = 1  $\wedge$  -,comp then
    hv  a24      LO                   ;   go to end fat;
    ck  -2      NQB                  ;   if bit(R, 34)  $\neq$  1  $\vee$  comp then
    hv  a25      NO                   ;   go to fat error ;
    pm  d25      DX                  ;   output(outrepr(<fat,>)); normal:= t;
    hv  e3      IZC                  ;   commok:= f; return;

a25:hs  c70                        ; fat error: alarm( $\{<\}$ <improper> $\}$ , 0+1);
    pm  (b1)    , qq se34            ;   R:= table[char];
    hv  a19      X                    ;   go to test sem;

```

```

c26:hsn c2      IZA      ; colon: normal:= f; call(next);
    ca d33[=]    ;      if part1(R) = repr(<=>) ^ -,comp then
    pm d30 DVX NQC ;      begin output(outrepr(<=>));
    pm d27 DVX    ;      normal:= t; commok:= f; return
    hv e3      IZC      ;      end;
    hs e3      IZC      ;      output(outrepr(†:>)); normal:= t;
    hv c3      X        ;      commok:= f; go to innormal;

c27:hsn c2      IZA      ; equal: normal:= f; call(next);
    ca d18[>]    ;      if part1(R) = repr(<>>) ^ -,comp then
    pm d31 DVX NQC ;      begin output(outrepr(<=>>));
    pm d28 DVX    ;      normal:= t; commok:= f; return
    hv e3      IZC      ;      end;
    hs e3      IZC      ;      output(outrepr(<=>)); normal:= t;
    hv c3      X        ;      commok:= f; go to innormal;

c28:hsn c2      IZA      ; minus: normal:= f; call(next);
    ca d16[, ]    ;      if part1(R) = repr(<,>) ^ -,comp then
    pm d32 DVX NQC ;      begin output(outrepr(<-,>));
    pm d29 DVX    ;      normal:= t; commok:= f; return
    hv e3      IZC      ;      end;
    hs e3      IZC      ;      output(outrepr(<->)); normal:= t;
    hv c3      X        ;      commok:= f; go to innormal;

c29:hs c70      ; error in normal: alarm(†<character†, 1);
c30:hv c2      , qq se31 ; space in normal: go to next;
c:             ; comment for redefinition of normal actions.

                ; procedure alarm(text, type);
c70:pmn 1b      X      IPC ; begin print:= f; R:=M:= 0
    hvn e5      ;      call(message); return
                ;      end alarm;
e              ;      end main block

```

[4.2. RESERVATION FOR INPUT TABLE UPPER CASE]

```

d2:             ;      input table upper case[0]
i=65i          ;      reserve input table upper case[0:64]

s

```

[4.3. STRING BLOCK]

b a40

[Constants and working locations used in layout and string:

	In layout,	basic state.	In string]
a1: qq	1.39;	minus word	[0] bit 39
a2: qq	2.39;	plus word	[0]
a3: qq	3.39;	lined plus word	[0]
a4: qq 1.4+	1.33+1.37;	d increment	[0]
a5: qq 1.4+	1.37;	z increment	[0] bit 37
a6: qq 1.4+1.23+1.33+	1.37;	n increment	[0]
b20:qq [fullword],		; layout word	str word (uses ,mark)
[layout word format:			
dhs check.4 or .3+d.22+n.23+s.26+fe.28+boverflow.29+b.33+h.37+fn.39]			
b21:qq [full word]		; space word	work 1
b22:qq [full word]		;	work 2
qq 1.4+1.22+1.33		; d increment	[6]
qq 1.4+1.22		; z increment	[6]
b23:qq [address only]		;	work 3
qq 1.28		; minus word	[12]
qq 2.28		; plus word	[12]
qq 3.28		; lined plus word	[12]
qq 1.3+1.26		; d increment	[12]
a7: qq 1.19		;	
a8: qq 10.3+10.9+10.15+10.21+10.27+10.33+10.39;			ends;
		; string or layout:	
		; <u>begin</u>	
c40:hsn c1	IZA	; normal:= f; call(next1);	
ca d15[<]		; <u>if</u> -,lined ^ part1(R) = repr(<<>)	
hv c52	NQB	; <u>then go to</u> string;	
pa b30 X	-19	; layout: position:= -19;	
grn b20 , gr b21		; layoutword:= spaceword:= 0;	
pp 0		; state:= 0;	
ps 0 X		; basic state:= 0;	
hv a14	NQB	; <u>if</u> compound <u>then</u>	
a10:ar a7	NQA	; comp layout symbol:	
gt r , pm -1		; <u>begin</u> R:= table[part2(R) + (<u>if</u> lined	
hv a14 X	LQA	; <u>then</u> 0 <u>else</u> 1)];	
pm d	NA	; <u>if</u> bit(R, 40) ≠ 1 ^ -, lined <u>then</u>	
hv a14 X		; R:= table[comp error];	
		; <u>go to</u> examine;	
		; d: <u>comment</u> allowed in states	
c41:qq 3.1+15.7+7.12+3.38		; 0,1,3,4,7,8,9,10,13,14,15;	
pm sa4		; M:= d increment[basic state];	
a11:pp s3 X		; n1: state:= basic state + 3; R:= M;	
a12:ac b20	IOA	; zerol: layout word:= layout word + R;	
b30:qq <u>-1</u> V	1 NOA	; <u>if</u> bit(layout word, 0) = 1 <u>then</u> state:= -1	
pp -1		; <u>else</u> position:= position + 1;	
a13:hsn c2	IZA	; next layout: callnext;	
hv a10	LQB	; <u>if</u> comp <u>then go to</u> comp layout symbol;	
a14:ck 20 , ga b31		; examine: layout type := part3(R);	
b31:pm <u>-1</u> X		; R:= table[layout type];	
ck p-3	IQC	; comp:= lined:= f; <u>if</u> bit(R, state-3) = 1	
hv (b31) t 1 LO		; <u>then go to</u> layoutaction[layout type + 1];	
pp -1 , ck 20		; state:= -1;	
hv (b31) t 1 LO		; <u>go to if</u> bit(R, state+15) = 1 <u>then</u>	
hv a13		; terminator <u>else</u> next layout;	

```

; zero: comment allowed in states
c42:qq 15.3+15.9 ; 3,4,5,6,9,10,11,12;
    pp s5 , pm sa5 ; R:= z increment[basic state];
    hv a12 X ; state:= basic state + 5; go to zero1;

c43:qq 3.38 ; n: comment allowed in states 0,1;
    pm a6 , hv a11 ; M:= n increment; go to n1;

c44:qq 3.38+5.2 ; point: comment allowed in states 0,1,3,5;
    qq (b30) t 1 ; position:= position + 1; state:= state + 7;
    pp p7 , ps 6 ; set adr[layout word, 0];
    pa b20 , hv a13 ; basic state:= 6; go to next layout;

c45:qq 5.2+5.8 ; ten: comment allowed in states 3,5,9,11;
    pp 13 , ps 12 ; state := 13; basic state 12;
    pa b20 , hv a13 ; set adr(layout word, 0);
    ; go to next layout;

c46:qq 5.2+5.8 ; space: comment allowed in states 3,5,9,11;
    pm 1.4 DX ; space word:= space word + bit(19 + position);
    ns (b30) , ck s-15 ;
    ac b21 , pp p1 ; state:= if position > 0 then -1
    bs (b30) , pp -1 ; else state + 1;
    hv a13 ; go to next layout;

c47:qq 1.10+1.37 ; minus: comment allowed in states 0,13;
    arn sa1 , ac b20 ; layout word:= layout word +
    pp p1 , hv a13 ; minusword[basic state];
    ; state:= state + 1; go to next layout;

c48:qq 1.10+1.37 ; plus: comment allowed in states 0,13;
    arn sa2 , ac b20 ; layout word:= layout word +
    pp p1 , hv a13 ; plusword[basic state];
    ; state:= state + 1; go to next layout;

c49:qq 1.10+1.37 ; lined plus: comment allowed in states 0,13;
    arn sa3 , ac b20 ; layout word:= layout word +
    pp p1 , hv a13 ; lined plus word[basic state];
    ; state:= state + 1; go to next layout;

c50:qq 5.2+5.8+1.11-1.36 ; terminator: comment allowed in states
    pm d26 DX ; 3,5,9,11,15 and in error;
    hs e3 IZC ; normal:= t; commok:= f;
    pm (b21) DX ; output(outrepr(<layout>));
    hs e3 ;
    pm b21 X IQC ; output(part1(space word));
    tk 10 , hs e3 ; comp:= lined:= f;
    pm b20 , cln -10 ; output(part2(space word));
    hs e3 , qq c2-1 ; output(part4(layout word));
    cln -1 , bs p1 ; if state = -1 v
    cln -10 V NO ; bit(layout word, 29) ≠ 0 then
    hs c70 ; alarm({<string>, 1);
    hv e3 , qq se36 ; output(bits(layout word, 19, 28));
    ; return;

c51:qq ; all others: comment only table word needed;

```



```

; begin string:
c52:pa b35 t -1 IRA ; str case:= -1; first word:= t;
pp d40 ; shift:= initial shift;
a16:hs c2 , qq i-1 ; str: call next; ch return(str);
hv a25 LQB ; if comp then go to comp str;
b35:
a17:ck -1 X IOB ; str1: set case:= bit(R, 40-str case) = 1; R:= M;
pm (b35) XD 61 LOB ; if set case then begin M:= R; str case:=
; str case + 61; set adr0(R, str case) end;
a18:bs p-35 , hv pd41 ; str2: if shift > 35 then go to
; if shift = 36 then newword else first char;
a19:ck p-26 , pp p6 ; str3: pack(str word, 30-shift, 35-shift,
ac b20 ; bits(R, 4, 9)); shift:= shift + 6;
hv s1 NOB ; if -m set case then return;
a20:ns (b35) , it s57 ; new case: str case:= 57 - str case;
pa b35 ; set case:= f; R:= M;
hv a18 X IZB ; go to str2;

a21:gm b21 LOB ; new word: if set case then work1:= M;
pm b20 , ck 8 ; M:= str word;
gr b20 , pa b20 ; pack(str word, 0,35,0, 36,39,bits(R, 6, 9));
a22: ; store word:
b39:bt -36 t 1 ; word count:= word count + 1;
hs a29 ; if word count > 1 then call(track full);
b36:pp -1 IK ; str adr:= str adr + 1;
b38:gm 2d43 t 1 MOC ; pack(str buf[str adr], 0,39,M,
pi 0 IK ; 40,41,str flags); shift:= 0;
ga b36 LZA ; if -,normal then str flags:= bits(R, 4,5);
hs a28 NRA ; if first word then call(output long);
hv s1 NOB ; if -,set case then return;
pm b21 , hv a20 ; M:= work1; go to new case;

a24:ck 8 , gr b20 ; first char:
pa b20 , ga b36 ; pack(str word, 0,35,0, 36,39,bits(R,6,9));
pp 0 ; str flags:= bits(R, 4, 5); shift:= 0;
hv s1 NOB ; if -,set case then return;
hv a20 ; go to new case;

a25:gm b23 , pm a5 ; comp str: work2:= M; M:= bit37;
hv a27 NQA ; if -,lined then go to str bar; lined:= comp:= f;
ca d16[, ]X IZB ; if part1(R) = repr(<,>) then
hvn a18 IQC ; begin set case:= f; R:= 0; go to str2 end;
a26:pmd17[_] D IQC ; single: R:= M; set adr0(M, charrepr(<_>));
hs a17 ; call(str1);
pm (b1) X ; R:= table[char]; M:= work2;
pm b23 , hr a17 ; go to str1;

```



```
e ; end type in;
```

[5. Tables]

[5.1. Formats]

[Pass 1 uses 3 different tables:

5.2. Programs for compound symbols not identifiable on first char. Contains explicit code pieces which (by help of the general mechanisms next and next 1) reads and identifies the necessary number of underlined letters for the compound symbols which can not be identified on first letter alone.
Entry points to these code pieces are given as compactions in compound table, determined by first lined or barred character.

5.4. Compound table:

This table contains one entry for each allowed compound symbol, for undefined compounds, for undefined bar compound, and for lined CR or SP.

Each entry consists of one word holding 3 bytes :

1. Output value or normal action.

(Output values < 512; normal actions > 511);

2. Compound action

3. Layout action

Each allowed bar compound is flagged with , mark. The table is referred either from the input table (for first lined or barred character) or from the codepieces 5.2. (for identified compound symbols)
Reference to a barred symbol is the reference to the same lined symbol + 1;

5.5. and 5.6. Input table lower and upper case.

Holds one word for each possible input character.

Totally forbidden symbols are flagged by the f mark missing and the code for pass 1 is regarded as part of the table for totally forbidden symbols.

The format of the tablewords proper is given in the table heading.

The byte combination bits holds the following booleans (1 \equiv t, 0 \equiv f);

Bit no	Dec. value	Meaning
39	1	Requires upper case in strings
38	2	May be fat and in fat; (letters, SP, CR, STOP, TAB)
37	4	Requires lower case in strings
36	8	Fat undetermined (SP, CR, STOP, TAB)
35	16	end fat (:);
34	32	finish fat (());
33	64	letter (used by comp error)
32	128	SP, CR (used by comp error)]

[5.2. PROGRAMS FOR COMPOUND SYMBOL NOT IDENTIFIABLE ON FIRST CHAR]

```

c60:hsn c2          IZA          ; b          normal:= f;  call(next);
      ca  5 [e], hv 10c15 ; begin      if Raddr=repr(<e>) then go to rest 3;
      pm 11d      , ca  15 [O];          if Raddr≠repr(<o>) then go to long red
      gm b        , hv  6c15 ; boolean                                comp err;
      hv c18              ;          comp symbol:= table[<boolean>];
                                   ;          go to rest 5;

c61:hsn c2          IZA          ; e
      ca 12 [l], hv 12c15 ; else
      pm 15d      , ca  14 [n];
      gm b        , hv 14c15 ; end
      hv c18              ;

c62:hsn c2          IZA          ; f
      ca  1 [a], hv 10c15 ; false
      pm 17d      , ca  15 [O];
      gm b        , hv 14c15 ; for
      hv c18              ;

c63:hsn c2          IZA          ; g
      hs c31      NQA          ;          if -, lined then call(comp rest error);
      hsn c1              ; go          call(next 1)
      ca d34[SP], hv 13c15 ; go to or go to
      ca 20 [t], hv 14c15 ; goto
      hv c18              ;

c64:hsn c2          IZA          ; i
      ca  6 [f], hv 16c15 ; if
      pm 20d[n], ca  14 [n];
      gm b        , hv  6c15 ; integer
      hv c18              ;

c65:hsn c2          IZA          ; s
      ca 20 [t], hv  c66      ; st
      pm 27d      , ca  23 [w];
      gm b        , hv  8c15 ; switch
      hv c18              ;

c66:hsn c1          LQC          ; st
      ca  5 [e], hv 14c15 ; step
      pm 26d      , ca  18 [r];
      gm b        , hv 10c15 ; string
      hv c18              ;

c67:hsn c2          IZA          ; t
      ca  8 [h], hv 12c15 ; then
      pm 29d      , ca  18 [r];
      gm b        , hv 12c15 ; true
      hv c18              ;

d38:qq 58.9+60.19+58.29+60.39;    constant four case shifts
d d39 = d38 - 1                  ;    define base (type buf);
d5:                               ;    top of pass 1

```

[5.3. REDEFINITION OF NORMAL ACTIONS]

```

d c20=c20-c, c21=c21-c, c22=c22-c, c24=c24-c, c25=c25-c
d c26=c26-c, c27=c27-c, c28=c28-c, c29=c29-c, c30=c30-c
d c2=c2-c , a=c51 ;
[Definition of normal action test values for semicolon, SP, and equal]
d d11=c21 ; semicolon
d d33=c27 ; equal
d d34=c30 ; SP

```

[5.4. COMPOUND TABLE]

[OUTPUT		COMP.-	LAYOUT	
CONST.		ACTION	ACTION	
or				
NORMAL				
ACTION				
NAME				COMPOUND SYMBOL]
d i=d				
qq	c18.19+	a.29		; undefined compound
qq	239.9+ c4.19+	a.29		; 1 ≡
qq	236.9+ c4.19+	a.29	,	; 2 ≠
qq	232.9+ c4.19+	a.29		; 3 ≤
qq	c40.19+	a.29	,	; 4 <
qq	234.9+ c4.19+	a.29		; 5 ≥
qq	c2.9+ c19.19+	c50.29	,	; 6 >
qq	c2.9+ c19.19+	a.29		; 7 △ (not used)
qq	229.9+ c4.19+	a.29	,	; 8 ↑
qq	131.9+ 9c15.19+	a.29		; 9 <u>array</u>
qq	c24.9+ c60.19+	a.29		; 10 <u>begin</u>
qq	117.9+ 5c15.19+	a.29		; 11 <u>boolean</u> or <u>Boolean</u>
qq	c20.9+ 5c15.19+	a.29		; 12 <u>comment</u>
qq	224.9+15c15.19+	a.29		; 13 <u>do</u>
qq	160.9+ c61.19+	a.29		; 14 <u>else</u>
qq	c22			; 15 <u>end</u>
qq	226.9+ c62.19+	a.29		; 16 <u>false</u>
qq	93			; 17 <u>for</u>
qq	87.9+ c63.19+	a.29		; 18 <u>go to</u> or <u>go to</u> or <u>goto</u>
qq	100.9+ c64.19+	a.29		; 19 <u>if</u>
qq	107			; 20 <u>integer</u>
qq	139.9+ 9c15.19+	a.29		; 21 <u>label</u>
qq	102.9+13c15.19+	a.29		; 22 <u>own</u>
qq	124.9+ 1c15.19+	a.29		; 23 <u>procedure</u>
qq	112.9+11c15.19+	a.29		; 24 <u>real</u>
qq	176.9+ c65.19+	a.29		; 25 <u>step</u>
qq	137			; 26 <u>string</u>
qq	135			; 27 <u>switch</u>
qq	167.9+ c67.19+	a.29		; 28 <u>then</u>
qq	225			; 29 <u>true</u>
qq	178.9+ 9c15.19+	a.29		; 30 <u>until</u>
qq	141.9+ 9c15.19+	a.29		; 31 <u>value</u>
qq	180.9+ 9c15.19+	a.29		; 32 <u>while</u>
qq	230.9+ c4.19+	a.29		; 33 <u>:</u>
qq	c2.9+ c19.19+	c46.29		; 34 <u>⌊</u>
qq	c2.9+ c19.19+	c49.29		; 35 <u>⊥</u>
qq	c2.9+ c17.19+	a.29		; 36 lined SP or CR
qq	c2.9+ c19.19+	a.29		; 37 undefined bar comp

[FORMATS

Relevant format
given by flags

ANYTHING WHAT SO EVER

no f-bit

hv SPECIAL
ACTION

f,

qq OUTPUT COMP.- LAYOUT COMBI- f
 CONST. TABLE- ACTION NATION
 or INDEX. (a is BITS
 NORMAL (d is error)
 ACTION error)
 NAME.

INPUT]

d i=d1

qq	c30.9+	36d.19+	c46.29+	138.39	f	;	0	SP
qq	58.9+	d.19+	a.29+	4.39	f	;	1	1
qq	59.9+	d.19+	a.29+	4.39	f	;	2	2
qq	60.9+	d.19+	a.29+	4.39	f	;	3	3
qq	61.9+	d.19+	a.29+	4.39	f	;	4	4
qq	62.9+	d.19+	a.29+	4.39	f	;	5	5
qq	63.9+	d.19+	a.29+	4.39	f	;	6	6
qq	64.9+	d.19+	a.29+	4.39	f	;	7	7
qq	65.9+	d.19+	a.29+	4.39	f	;	8	8
qq	66.9+	d.19+	a.29+	4.39	f	;	9	9
qq						;	10	used internally
qq	c2.9+	d.19+	a.29+	10.39	f	;	11	STOP
hv	c13				f,	;	12	END
qq	c29.9+	d.19+	a.29+	4.39	f	;	13	aa
hv	c8.9 +			4.39	f,	;	14	_
qq						;	15	un
qq	57.9+	d.19+	c42.29+	4.39	f	;	16	0
qq	231.9+	3d.19+	a.29+	4.39	f	;	17	<
qq	19.9+	25d.19+	a.29+	70.39	f	;	18	s
qq	20.9+	28d.19+	a.29+	70.39	f	;	19	t
qq	21.9+	30d.19+	a.29+	70.39	f	;	20	u
qq	22.9+	31d.19+	a.29+	6.39	f	;	21	v
qq	23.9+	32d.19+	a.29+	70.39	f	;	22	w
qq	24.9+	d.19+	a.29+	6.39	f	;	23	x
qq	25.9+	d.19+	a.29+	70.39	f	;	24	y
qq	26.9+	d.19+	a.29+	6.39	f	;	25	z
qq						;	26	un
qq	200.9+	34d.19+	a.29+	4.39	f	;	27	,
hv	c16				f,	;	28	CLEAR
qq	c29.9+	d.19+	a.29+	0.39	f	;	29	RED
qq	c2.9+	d.19+	a.29+	10.39	f	;	30	TAB
hv	c11				f,	;	31	POFF
qq	c28.9+	d.19+	c47.29+	4.39	f	;	32	-
qq	10.9+	d.19+	a.29+	6.39	f	;	33	j
qq	11.9+	d.19+	a.29+	6.39	f	;	34	k
qq	12.9+	21d.19+	a.29+	70.39	f	;	35	l
qq	13.9+	d.19+	a.29+	70.39	f	;	36	m
qq	14.9+	d.19+	c43.29+	70.39	f	;	37	n
qq	15.9+	22d.19+	a.29+	70.39	f	;	38	o
qq	16.9+	23d.19+	a.29+	70.39	f	;	39	p
qq	17.9+	d.19+	a.29+	6.39	f	;	40	q
qq	18.9+	24d.19+	a.29+	70.39	f	;	41	r

[5.5 INPUT TABLE lower case
continued

]

qq						; 42 un
qq	28.9+	d.19+	a.29+	6.39	f	; 43 ø
qq	c12				f,	; 44 PON
qq						; 45 un
qq						; 46 un
qq						; 47 un
qq	27.9+	d.19+	a.29+	6.39	f	; 48 æ
qq	1.9+	9d.19+	a.29+	70.39	f	; 49 a
qq	2.9+	10d.19+	a.29+	70.39	f	; 50 b
qq	3.9+	12d.19+	a.29+	70.39	f	; 51 c
qq	4.9+	13d.19+	c41.29+	70.39	f	; 52 d
qq	5.9+	14d.19+	a.29+	70.39	f	; 53 e
qq	6.9+	16d.19+	a.29+	6.39	f	; 54 f
qq	7.9+	18d.19+	a.29+	70.39	f	; 55 g
qq	8.9+	d.19+	a.29+	70.39	f	; 56 h
qq	9.9+	19d.19+	a.29+	70.39	f	; 57 i
hv	c6				f,	; 58 LC
qq	67.9+	d.19+	c44.29+	4.39	f	; 59 .
hv	c7				f,	; 60 UC
hv	c14				f,	; 61 SUM
qq	c29.9+	d.19+	a.29+	0.39	f	; 62 BLACK
qq						; 63 RF
d3:						
hv	c10				f,	; 64 CR
qq	c2.9+	d.19+	a.29+	4.39		; 65 _ secondary
qq	c2.9+	d.19+	a.29+	1.39		; 66 secondary
qq	c2.9+	36d.19+	a.29+	138.39		; 67 CR secondary

[5.6. INPUT TABLE upper case]

[FORMATS

Relevant format
given by flags

ANYTHING WHAT SO EVER

no f-bit

hv SPECIAL
ACTION

f,

qq OUTPUT COMP.- LAYOUT COMBI- f
 CONST. TABLE- ACTION NATION
 or INDEX. (a is BITS
 NORMAL (d is error)
 ACTION error)
 NAME.

INPUT]

d i=d2

qq	c30.9+	36d.19+	c46.29+	138.39	f	;	0	SP
qq	238.9+	d.19+	a.29+	1.39	f	;	1	✓
qq	227.9+	d.19+	a.29+	1.39	f	;	2	×
qq	228.9+	d.19+	a.29+	1.39	f	;	3	/
qq	c27.9+	1d.19+	a.29+	1.39	f	;	4	=
qq	c21.9+	d.19+	a.29+	1.39	f	;	5	;
qq	190.9+	d.19+	a.29+	1.39	f	;	6	[
qq	182.9+	d.19+	a.29+	1.39	f	;	7]
qq	165.9+	d.19+	a.29+	33.39	f	;	8	(
qq	c25.9+	d.19+	a.29+	1.39	f	;	9)
qq							;	10 un
qq	c2.9+	d.19+	a.29+	10.39	f	;	11	STOP
hv	c13				f,	;	12	END
qq	c29.9+	d.19+	a.29+	1.39	f	;	13	AA
hv	c9.9+			1.39	f,	;	14	
qq							;	15 un
qq	237.9+	7d.19+	a.29+	1.39	f	;	16	^
qq	235.9+	5d.19+	a.29+	1.39	f	;	17	>
qq	47.9+	d.19+	a.29+	3.39	f	;	18	S
qq	48.9+	d.19+	a.29+	3.39	f	;	19	T
qq	49.9+	d.19+	a.29+	3.39	f	;	20	U
qq	50.9+	d.19+	a.29+	3.39	f	;	21	V
qq	51.9+	d.19+	a.29+	3.39	f	;	22	W
qq	52.9+	d.19+	a.29+	3.39	f	;	23	X
qq	53.9+	d.19+	a.29+	3.39	f	;	24	Y
qq	54.9+	d.19+	a.29+	3.39	f	;	25	Z
qq							;	26 un
qq	68.9+	d.19+	c45.29+	1.39	f	;	27	¹⁰
hv	c16				f,	;	28	CLEAR
qq	c29.9+	d.19+	a.29+	0.39	f	;	29	RED
qq	c2.9+	d.19+	a.29+	10.39	f	;	30	TAB
hv	c11				f,	;	31	POFF
qq	72.9+	35d.19+	c48.29+	1.39	f	;	32	+
qq	38.9+	d.19+	a.29+	3.39	f	;	33	J
qq	39.9+	d.19+	a.29+	3.39	f	;	34	K
qq	40.9+	d.19+	a.29+	3.39	f	;	35	L
qq	41.9+	d.19+	a.29+	3.39	f	;	36	M
qq	42.9+	d.19+	a.29+	3.39	f	;	37	N
qq	43.9+	d.19+	a.29+	3.39	f	;	38	O
qq	44.9+	d.19+	a.29+	3.39	f	;	39	P
qq	45.9+	d.19+	a.29+	3.39	f	;	40	Q
qq	46.9+	d.19+	a.29+	3.39	f	;	41	R

[5.6 INPUT TABLE upper case
continued]

qq						; 42 un
qq	56.9+	d.19+	a.29+	3.39	f	; 43 Ø
hv	c12				f,	; 44 PON
qq						; 45 un
qq						; 46 un
qq						; 47 un
qq	55.9+	d.19+	a.29+	3.39	f	; 48 Æ
qq	29.9+	d.19+	a.29+	3.39	f	; 49 A
qq	30.9+	11d.19+	a.29+	3.39	f	; 50 B
qq	31.9+	d.19+	a.29+	3.39	f	; 51 C
qq	32.9+	d.19+	a.29+	3.39	f	; 52 D
qq	33.9+	d.19+	a.29+	3.39	f	; 53 E
qq	34.9+	d.19+	a.29+	3.39	f	; 54 F
qq	35.9+	d.19+	a.29+	3.39	f	; 55 G
qq	36.9+	d.19+	a.29+	3.39	f	; 56 H
qq	37.9+	d.19+	a.29+	3.39	f	; 57 I
hv	c6				f,	; 58 LC
qq	c26.9+	33d.19+	a.29+	17.39	f	; 59 :
hv	c7				f,	; 60 UC
hv	c14				f,	; 61 SUM
qq	c29.9+	d.19+	a.29+	0.39	f	; 62 BLACK
qq						; 63 TF
hv	c10				f,	; 64 CR

[6. END OF TAPE]

```

d e18 = c74           ; start pass 1
d i = 39d5           ; i:= top pass 1+39
d e16 = k - e70       ; e16:= first free track
d e22 = e16 - e15     ; e22 := number of tracks

```

```

e                     ; end pass 1

```

```

s

```

[26.2.65]

[Translator Loading Administration. Page 1]

```
d e84=e16 ; First free track after translator proper
d e19=e84+e86 ; Reserve e86 tracks between translator and output
d e87=e84-c61 ; Number of tracks used for translator
```

[Universal Translatorparameters. Rest of track 3e14]

```
d i=e4 ;

e4: qq e15 ; CR counter, from start: first track of pass 1
[1] qq 0 , pp c61 ; used tracks , first track of RS
[2] qq 0 , pp e0 ; inf 1 , - - - std. procs.
; HP entry: f mark = true, -, f mark = false;
[3] qq 0 , pp e24 ; inf 2 , - - - passes
[4] qq e20 , pp e81 ; last track , - - - std. tables
[5] qq e20-e19, ps e71 ; available tracks , number of std. identifiers
[6] qq e19 , pp e20 ; input track , last track for output
[7] qq e19-1 , qq -1 ; output track , string no
[8] it 1.4+1.5, vy 1.5+1.9; normal by , typewr by
[9] qq 1 , vy 1.5 ; pass no , error by
[10]qq e19-1 , bs 0 ; first track-1 , pass inf: ≤ 0 = false
; 0 4 true
```

[The following sections, up to but not including initialize translator,
are only used immediately after a SLIP-loading and are not output on
the compressed translator tape]

[Program for placing and moving translator]

```
d i=e89 ; first free after identifierlist
```

```
e27:vk 1e14+e70 , sk e13-39 ; forward pass
vk 2e14+e70 , sk 1e13 ; backward pass
bs e96 ; if HP entry then
acn 2e4 MC ; set f mark on 2e4;
vk 3e14+e70 , sk 165e13 ; printing and new line
vk 4e14+e70 , sk 124e13 ; endpass
vk c64+e70+1 , sk c67 ; running system 6
d i=i-e69 ;
vk c66+e70 , sk c67-40 ; running system 7
vk c64+e70+3 , sk c67-80 ; running system 8
d i=i-e69 ;
vk e70 , hh r7 ; move translator e70 tracks, e70>0
bs 1e70 , hv r9 ; e70=0
ps r5 , pp e84+e70-1; e70<0
vk p , lk e25 ;
vk p-e70 , sk e25 ;
pp p-1 , hv s-1 ;
bs p-c70+1 , hv r-3 ;
hv r3 , pp c70-2 ; e70>0
bsp-e84-e70+3, hv r2 ;
pp p2 , hs r-6 ;
can e96 , hv r5 ; if HP entry then
vk 24c61 , lk e25 ; begin set on track 25 (std.procs.
vk 24c61 , it e84-31c61-1; entry to algol from HP):
pa e25 , it e14 ; number of tracks to be summed,
pa 1e25 , sk e25 ; init. transl. track end
d i=i-e69-e69-e69-e69-e69 ;
vk ; wait for drum
```

[TLA. Page 2]

[Generation of standard identifier tables]

b a15, b18, c2, d8 ;

[This code builds up the tables as described in pass 2 from the list stored in core from 205e13.

This list holds for each identifier two or more words loaded during loading of the standard procedures.

The format of the words (running index *i* list) is:

```
ilist:  Pass 6 word for std identifier. All 42 bits.
        +1:  Name of identifier in slip t representation, as
        ---  many words as necessary.
```

It is assumed that none of the tables will take up more than 2 tracks.

The tables are formed totally in core before they are transferred to drum]

[Definitions of table area and initial values for indices]

```
d d1=165e13;idecl for decl[80]           ; decl[0:79];
d d2= 1e13; base for buf long           ; buf long[0:79];
d d3= 40e25; i buf short for buf short[80] ; buf short[0:79];
d d4=d3-137; base for letter;           ; letter[0:56]
d d5=e91+14; e91= address of chain start list[0] from pass 2
        ; d5 = initial i long;
d d6=d3      ; i short, initial value;
d d7=204e13; i list for list[-1]; list[0:e89-1];
d d8=14d2    ; i buf long for buf long[14];
```

[Other initial values:

```
ident count=e71, short tracks=e29, long tracks=e88,
first table track =e81]
```

```

c:  pp  57      , pp  p-1      ; initialize letter table:
    grn pd4      , M           ;   for p:= 56 step -1 until 1 do
    it  pd4      , pt  pd4      ;   pack(letter[p], 0, 9, 0,
    bs  p-1      , hh  c        ;   10, 19, p+baseletter, 20, 39, 0);
    ps  d5       , hv  a10      ;   s:= initial i long; go to count ident;
b18:qq  0        t  e0         ; first track of std procs with unit in pos 19
d i = i - e69      ; skip for CDC
c1:               ;
b1:  pmn d7       X  1  IQC     ; start ident: i list:= ilist + 1 ; idecl:= idecl + 1;
    sr  b18       NA          ;
d i=i-e69          ; CDC: skip previous
b2:  gr  d1       t  -1 MQC     ;   pack(decl[idecl], 0, 41, list[ilist] - b18);
    pa  b5        , gm  b6      ;   case:= work:= 0;

a1:  pmn(b1)     X  1          ; next char word:
    cl  36        ;   i list:= i list + 1; R:= 0;
    ar  32        D    LA      ;   M:= bits(0, 35, list[i list]);
    ar  16        D    LB      ;   pack(R, 0, 5, bits(36, 41, list[i list]));
    pa  b3        , hv  a2      ;   count:= 6; go to first char;

c2:  gr  b6       , it  -80     ; get char: work:= R;
b3:  bt  -1       , hv  a1      ; space: count:= count - 1;
    pmn b7       , cl  -6      ;   if count < 0 then go to next char word;
    ;   M:= bits(0, 33, charword); R:= 0;
    ;   pack(R, 0, 5, bits(34, 39, charword));
a2:  ck  -4       , ga  b4      ; first char: Raddr:= p:= bits(0, 5. R);
b4:  pp  -1       , gm  b7      ;   charword:= M; if Raddr = 0 then go to space;
    hh  c2        , LZ         ;   if Raddr = <end text> then
    ca  10        , hvn a15     ;   begin R:= 0; return end;
    ca  58[LC],   hh  a3        ;   if Raddr = <LC> then go to LC;
    ca  60[UC],   hv  a3        ;   if Raddr = <UC> then go to UC;
    ca  16[0] ,   pp  0         ;   if Raddr = <0> then p:= 0;
    ca  43[Ø] ,   pp  27        ;   if Raddr = <Ø> then p:= 27;
    ca  48[æ] ,   pp  26        ;   if Raddr = <æ> then p:= 26;
    bs  p-48     , pp  p-105    ;   if p > 48 then p:= p - 105;
    bs  p-32     , pp  p-80     ;   if p > 32 then p:= p - 80;
    bs  p-17     , pp  p-56     ;   if p > 17 then p:= p - 56;
b5:  pp  p-1     , pp  p57      ;   p:= p + case + 57;
    pm  p        DX           ;   R:= 0; Raddr:= p; M:= work
a15:pm  b6       , hr  s1      ;   finis:= LA:= true; exit;

a3:  it  60       , pa  b5      ; UC: case:= 28; go to space;
    hh  c2        ; LC: case:= 0; go to space;

; comment variables and constants:
b6:  qq          ;   work. no .mark used for finis:= true;
b7:  qq          ;   char word
b8:  qq  67.39,   ;   67. ,mark used for finis:= false;
b9:  qq e88.19-1.19+e29.29+d6.39-e71.39-513.39 ; std ident. inf.word
[long track -1.19+ short tracks.29+first free ishort-512.39];

```

```

a6: hsn c1      IZC      ; next identifier:
    arn pd4    , ck 10   ;   ZC:= 3; short:= t; call(start ident);
    ga b11     , arn(b11);   i short := i short - 1; i buf short:= i buf short -
b10: ar d6     D  -1  IPA ;   pack(store[part 2(letter[p]),
b11: gr -1     MPA      ;       0, 9, i short, 41, 41, 0); R:= 0;
b12: it d3     t  -1    ;   pack(letter[p], 10, 19, i buf short);
    ptn pd4    ;
                                ; next char: call(Heet ahar);
a7:  hs c2      ;   if Raddr = 0 then
    hv a9      X  LZ    ;   begin R:= RM; go to end ident end;
    ck 10      , ml b8   ;   RM:= RM×67 + Raddr; finis:= f;
    hv a7      X  LZ    ;   if RM < 2↑39 then word not full:
                                ;   begin R:= RM; go to next char end;
b13: gm d8     V  1  NZA ;   if -, short then storelong:
                                ;   begin i buf long:= i buf long + 1;
a8:  cl 19     XV    IZA ;   buf long[i buf long]:= remain(RM, 2↑39);
                                ;   R:= RM:2↑39; s:= s + 1; go to next char
    ps s1      , hv a7   ;   end;
                                ; store first long: a8: ZC:= ZC - 2; short:= f;
    ck -19     ;   pack(buf short[i buf short], 0, 9, 0, 10, 19, s,
    gr (b12)  X    MZC   ;   20, 39, remain(RM:2↑39), 40, 41, ZC);
    it s      , pt (b12) ;   R:= RM:2↑20; if -, finis then go to next char;
    hv a7     LA      ;
a9:  nc 0      XV    LZA ; end ident:
    ar 512     DV      ;   if short then try store short;
    hvn a8     ;   begin RM:= R; if Raddr ≠ 0 then
                                ;   begin R:= 0; go to store first long end;
    gr (b13)  V  1  NZA ;   pack(bufshort[i buf short], 0, 39, RM;
                                ;   40, 41, ZC)
    gm (b12)  V    MZC   ;   end
                                ; else store last of long:
    ps s1     ;   begin i buf long:= i buf long + 1; s:= s + 1;
                                ;   buf long[i buf long]:= R ++2↑39
a10:         ;   end;
b14: bt e71    t  -1    ; count ident: ident count:= ident count - 1;
    hv a6      ;   if ident count ≥ 0 then go to next identifier;

```

```

a11:pm b9      , gm d2      ; pack letter table in long buf:
    gs d2      , pp 0      ;   pack(long buf[0], 0, 9, s,
a12:psn -1     , pp p1     ;           10, 19, long tracks - 1,
    ar (pd4) D  ;           20, 29, short tracks,
    ck 10      , ps s10    ;           30, 39, i short - 512);
    bs s480    , hh a12    ;   p:= 0;
b15:gr d2      t 1 M      ;   for i:= 1 step 1 until 14 do
    bs p456    , hv a12    ;       for s:= 9 step 10 until 39 do
    gr (b15)   MA        ;       begin p:= p + 1; pack(long buf[i],
a13:          ;       s-9, s, bits(0, 9, lettertable[p]) end;
b16:vk e81-1   t 1      ;   tables to drum:
b17:sk d2-40   t 40     ;       track:= track no of first table track - 1;
    bt e88-1   t -1     ;       i:= -40;
    hv a13     ;       for j:= 1 step 1 until long tracks do
    it (b17)   t 39     ;       begin track:= track + 1; i:= i + 40;
    bs (b13)   , hs e5   ;       to drum(track, long buf[i]) end;
    hv r1      , qq (e30);   if i + 39 < i long buf then
a14:vk (b16)   t 1      ;       alarm({<program too big>, 4 + 0);
    sk d3      t -40    ;       i:= 0;
    is (b16)   , vk se29 ;       for j:= 1 step 1 until short tracks do
    sk d1      t -40    ;       begin track:= track + 1; i:= i - 40;
    bt e29-1   t -1     ;       to drum(track, short buf(top short - i));
    hv a14     ;       to drum(track + short tracks,
a14:          ;       decl[top decl - 40])
    e          ;       end
    b k=36, i=0 ;   end initialize tables.
    d i=11     ;   h algol to hjælp
    h algol/e95.29+c66 ;
    e          ;

```

[Initialize translator. Track e14]

```

b k=e14+e70, a5, b7      ;

a:  grn 1023 , vk 3e14 ; initialize core: out of hjælp;
    lk 165e13 , pm rb7 ; to core(print track, place for print track);
a1:                                ; store[1]:= jump to initialize translator;
b1: grn 165e13t -1 M ;
    bs (rb1) t 1e13 ; clear marks on input output buffers;
    gm 1 , hv ra1 ;
a2: grne13-41 t 82 MA ; set marks on bufferlimits;
a3: grn e13 t 82 MC ;
    vk 1e14 , ud ra2 ; to core(forward, in out place);
    lk e13-39 , ud ra3 ;

    ps (e4) , pp e22 ; pass 1 to core: track:= first pass 1 track -1;
a4: vk s-1 t 1 ; core:= pass place - 40;
    lk 165e13 t 40 ; for p:= pass 1 tracks step -1 until 1 do
    pp p-1 , bs p ; begin track:= track+1; core:= core+40;
    pa e4 , hv ra4 ; to core(track, core); CR counter:= 0 end;
                                ; ready: to core (mess track, place for mess);
    vk 5e14 , lk 124e13 ; tk:= first text track;
    vk 6e14 , grn 133e13; set no new line; message(if NKC then <
    pt rb3 t e62 NKA ; algol
    pt rb3 t e59 NKB ; < else if NKA then <
    pt rb3 t e58 NKC ; algol KB.
b2: qq -1 , hs 126e13 ; < else if NKB then <
b3: hv r1 ,qqe63+10.29; algol KA.
                                ; < else <
                                ; algol KC.
                                ; <, 8+2);
                                ; operational choice: LZ:= false;
[initial values: i=a=p=w=t=o=l=n=false]
a5: arn rb2 , ud 8e4 ; by:= typewr by; char:= read a char;
    ly rb2 , ca 57[i] ; if char=<i> then pass inf:= LZ:= true;
    ptn 10e4 t 1 ;
    ca 49[a] , hv e93 ; if char=<a> then go to init transl;
    ca 39[p] , pan rb4 ; if char=<p> then p:= LZ:= true;
    ca 22[w] , ptn rb4 ; if char=<w> then w:= LZ:= true;
    ca 19[t] , pan rb5 ; if char=<t> then t:= LZ:= true;
    ca 38[o] , ptn rb5 ; if char=<o> then o:= LZ:= true;
    ca 35[l] , pan rb6 ; if char=<l> then l:= LZ:= true;
    ca 37[n] , ptn 9e4 ; if char=<n> then n:= LZ:= true;
    hv ra5 LZ ; if LZ then go to operational choice;
                                ; set the choices: normal by:= normal by -
b4: nt 1.4 t 1.5 ; (if-,p then bit4 else 0)-(if-,w then bit5 else0);
    arn(8e4) D ; error by:= if n then normal by else
    ck -10 , ab 9e4 ; combined(normal by, typewr by);
    gt 9e4 , ppl.0+1.2+1.4+1.5; set initial indicator for pass 1;
b5: pp p+1.9 , pp p+1.3 ; line print:= 1; ignore POFF:= o;
b6: pi p-1.2 , vy (8e4) ; type input := t; by:= normal by;
    hv e18 ; go to start pass 1;

b7: hv e93 ; jump to init transl;

e ; end initialize translator;

```


[Formation of checksums]

```

b a5, b2, d3 ;
d d1=e25, d2=d1-40 ; Define buffer 1 and buffer 2
d d3=e80-c67+d1+160 ; - address of drumplace when track c61 in d1
hv a ; go to summations
b: qq 688.9+e95.19+812.29; h algol word;
a1: lk (b1) , pp 40 ; procedure sum; comment from track given by
vk (s) t 1 ; vk instruction in cell[s]. number of tracks -1
it (b1) , pt b2 ; given as address in cell[s+1].
b1: nt d1 , lk d1+d2 ; Track 38 is skipped during summation:
is (s) , it s473 ;
bs -511 , hv a3 ;
b2: pp p-1 , ar p-1 ;
bs p , hv r-1 ;
a3: bt (s1) t -1 ;
hh a1 ;
hh s1 ; exit;
a: vk c66 , hsn a1 ; sum (track c66);
qq 0 ;
vk c66 , lk d1 ; modify sum with h algol word and set in cell 39
vk c66 , ar b ; on track c66;
sc 39d1 , sk d1 ;
vk 0 , hsn a1 ; sum (track 0);
qq 0 ;
vk c61 , hs a1 ; add sum (first 31 tracks of compiler);
qq 30 , vk c61 ;
lk d1 , vk c61 ; set sum in drumplace
sc d3 IOA ;
pa d3-e80+c46t512LOA ; if overflow then set bit 0 in last used and
pa d3-e80+c19t512LOA ; start of program to correct for overflow;
sk d1 ;
vk c61 , hsn a1 ; sum (up to first track after std procs );
qq e98-c61-1, vke98-1;
lk d1 , vk e98-1 ; set sum in last cell of last track
sc 39 d1 , sk d1 ; of std procs;
vk e81 , hsn a1 ; sum(from first std table track to last pass track);
qq e84-e81-1, vk 7e14 ;
d a5=e69+e69, a5=a5+a5+e69; a5:= 5xe69
d i=i-a5-a5-a5 ; skip previous 15xe69 instructions
vk c61 , hsn a1 ; CDC summation
qq e87-1 , vk 7e14 ;
d i=i+e69+e69-2 ;
lk d1 , vk 7e14 ; set sum in last word of last text track;
sc 39d1 , sk d1 ;

a4: hv e93 ; end sums: go to initialize translator;
b k=c61+e70, i=0 ; Information about
d i=a5+a5+a5+a5+a5 ;
qq e97.9+e20.19+e84.29-1.29+e98.39-1.39 ; translator storing
qq c89.9+c29.19+c19.29+c55.39 ; to track c61
qq e14.9 ;
e
e

b k=5e0+e70+e96, i=0 ; word to to drum
d i=23
pm e98 DX ;
e ; Stop for possible reading of Kompud of passes
s ; or Kompud of pass 2 through 8
e e27 ; END TRANSLATOR, goto e27
s ;

```